Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.



THIS SPEC IS OBSOLETE

Spec No: 002-04944

Spec Title: AN204944 - DCF77-time signals with the MB90670/5 series by using the ICU

Replaced by: NONE

AN204944

DCF77-time signals with the MB90670/5 Series by using the ICU

This application note shows how to connect a DCF77 receiver to a MB90678 using the input capture function.

Contents

1 Introduction.....1
2 Document History.....15

1 Introduction

In many applications, exact time information is essential. Using a LF receiver for the German time standard DCF77 can be a simple, but effective solution for this problem. But how to decode the time-signal? This application note shows how to connect a DCF77 receiver to a MB90678 using the input capture function.

The 1978 law on time standards defines legal time in Europe on the basis of "Coordinated World Time (UTC)" and gives the PTB (Physikalisch-technische Bundesanstalt, Germany) responsibility for the keeping and broadcasting of legal time. Legal time in Europe is either Middle European Time (Mitteleuropäische Zeit) : MEZ(D) = UTC(PTB) + 1h or Middle European Summer Time MESZ(D) = UTC(PTB) + 2h and is generated in the PTB Atomic Clock in Braunschweig. It is transmitted from a 50kW-transmitter in Mainflingen (near Frankfurt) on the frequency 77.5 kHz with a relative deviation of the carrier (averaged over a day) of $< 1^{-12}$. The signal is transmitted in a 24-hour continuous service. Short interruptions (of a few minutes) are possible because of servicing, when switching to a backup transmitter or antenna. Thunderstorms can cause longer interruptions to the service.

The carrier is amplitude-modulated with second marks. At the beginning of each second (with the exception of the 59th second of each minute), the carrier amplitude is reduced to 25% for the duration of either 100 or 200ms. The start of the carrier reduction marks the precise beginning of the second. The minute is marked by the absence of the previous second mark. It is possible to achieve accuracy better than 1ms at distances of several hundred kilometers around Frankfurt.

The transmission of the numerical values for minute, hour, day, weekday, month and year are BCD-encoded through the pulse duration modulation of the second marks. A second mark with duration 0.1s encodes a binary 0 and a duration of 0.2s encodes 1. The order of encoding is shown in Table 1. In the case of transmission of MEZ, mark 18 has a duration of 0.2s and mark 17 a duration of 0.1s. If MESZ is being transmitted, this is reversed. Furthermore, an approaching transition from MEZ to MESZ or back is announced by extending mark 16 from 0.1 to 0.2s for one hour prior to the changeover.

OBSOLETE

Table 1. Encoding table

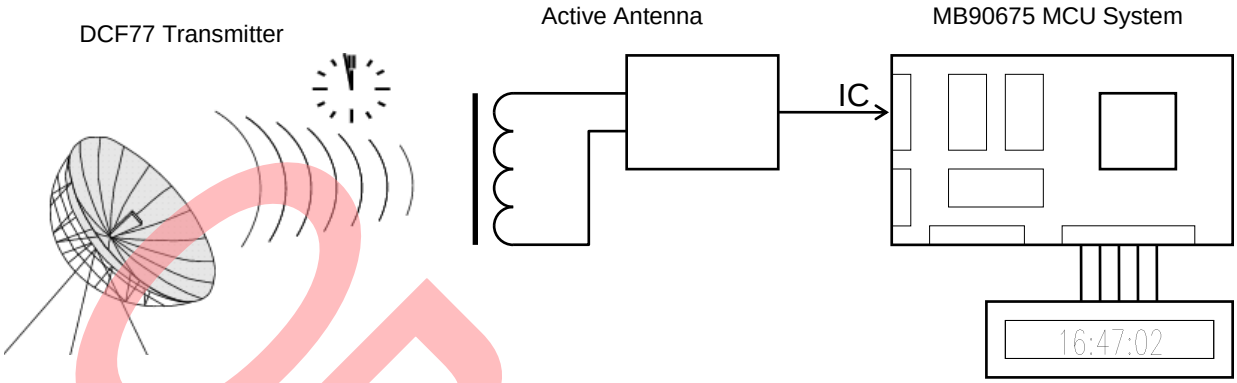
Mark number	Encodes (0.1s = logic “0” ; 0.2s = logic “1”)
0	Minute, always 0
1-14	Reserved
15	0=Normal antenna, 1=backup antenna in use
16	1=Approaching change from MEZ to MESZ or back (1 hour before)
17,18	Time zone 0,1=MEZ; 1,0=MESZ
19	The leap second is encoded in this bit one hour prior to occurrence.
20	Start bit for encoded time, always 1
21-27	1, 2, 4, 8, 10, 20, 40 Minutes
28	P1 maintains even parity for marks 21-28
29-34	1,2,4,8,10,20 Hours
35	P2 maintains even parity for marks 29-35
36-41	Day in month (1, 2, 4, 8, 10, 20)
42-44	Day in week (1,2,4)
45-49	Month number (1, 2, 4, 8, 10)
50-57	Year (1, 2, 4, 8, 10, 20, 40, 80)
58	P3 maintains even parity for marks 36-58 There is no mark transmitted for the 59th second.

To obtain the code, any active antenna capable of receiving 77kHz and demodulating the signal can be used. Such devices can be purchased by a number of electronic distributors in Europe. To feed the signal into the microcontroller, the input capture unit (ICU) of the MB90670/5 series is purpose-build. This module detects rising or falling edges, measures timeslots and generates interrupts. The maximum capture accuracy is 250 ns.

Figure 1 shows the example application using a FZD01020 active antenna connected to the input capture pin ASR1 of the MCU. The written demo software drives a dot matrix display connected to port 6 (see last issue of spectrum!) to show the actual time and date. Additionally the MCU sends the time-information via RS232 to a connected PC once the program is running. If the signal will not be received anymore, an internal timer (16-bit timer0) will continue to count the seconds up until the DCF77-signal is available again.

OBSOLETE

Figure 1. A simple time processing system using the MB90675



OBSOLETE

A system like this has many advantages compared to a normal quartz-based clock :

- Very accurate time (+/- 1 second in 1 Mio. years!)
- Can be a time base for a whole computer network
- automatic recognition of MEZ and MESZ
- Automatic recognition of leap years and minutes (!!!)
- Easy to program accurate alarms over month

The input capture unit can be set up easily and then works independently from the CPU, so that a “decode-DFC77”-task can be run almost in the background of a main MCU-program. The MB90670/75 series feature four input capture units, so other applications needing edge detecting can be run simultaneously. For further information (democode etc.) please contact the MCU-group in FMG.

```
/*
****************************************************************
**                                                            **
**                    DFC77-CLOCK                             **
**                                                            **
**  for MB90678 (16bit-Starterkit)                        **
**  can be run with Emulator                              **
**                                                  **
**  reads DFC77-Signal-Pulses via ICU1 (ASR1-Pin)         **
**  decodes the time signal and displays it on a LCD      **
**  connected to Port 6 (default-LCD-Type : LTN111)       **
**  (active antenna is FZD01020 by Channel Microelectronics     **
**  don't forget to add a pullup-restistor for signal-output    **
**  of antenna - it's an open-drain type)                       **
**                                                  **
**  Author : Markus Mierse                                **
**  Version : 1 (backup timer for "no-signal"-case disabled)    **
**  Date   : 17.2.97                                          **
**                                                            **
**  (C) Fujitsu Mikroelektronik GmbH 1997                     **
**                                                            **
****************************************************************
*/
```

```
#include <sample\extn\mb90675.h> /* for all Register Names */


/*----------------------- variables --------------------*/


typedef unsigned char BYTE;       /* BYTE definition */

int cursor;                 /* display position */

int i;              /* general purpose integer variable */
int counter;                /* counter for timer 0 */
int index;                  /* index (0-58) of time protocol */
int begin;                  /* 1 if begun to decode */

int min;            /* update time and date variables */
int hr;
int day;
int dow;
int month;
int year;

int sec;            /* actual time and date */
int minr;
int hrr;
int dayr;
int dowr;
int monthr;
int yearr;

long int puls;                  /* pulswidth of received signal */
long int dist;                  /* distance from last puls */
long int t1;                /* pulsvariable */


/*---------------------  prototypes --------------------*/

void   Init_Timer16(void); /* init timer routine */
void   Init_ICU(void);     /* init ICU */
void    decode(void);               /* decode-routine */
void    Init_Timer24(void);         /* init 24-bit-timer routine */

__interrupt
void    Timer16(void);              /* ISR for timer0 */

__interrupt
void    Timer24of(void);    /* ISR for 24-bit timer overflow */

__interrupt
void   ICU1(void);          /* ISR for ICU1 */

void initdisp(void);                /* display routines */
void outb(unsigned char);
void busy(void);
void print(char *Name2);
void printnum(int n);
void printtime(int h,int m,int s);
void printdate(int dw,int dr,int mr, int yr);
void wait(int i);
```

OBSOLETE

```
/*---------------------------------------------------------------------*/
/*            MAIN PROGRAM                     */
/*---------------------------------------------------------------------*/

void main(void)
{

  initdisp();               /* setup display on port 6 */
  print("DFC77-TIMER");     /* show header */

/*  Init_Timer16();         /* init 16-bit timer 0 (disabled) */

   Init_Timer24();          /* init 24-bit timer */
   Init_ICU();                   /* init ICU */

   dist = 0;                     /* init variables */
   begin = 0;
   i = 0;
   minr = 0;
   hrr = 0;
   sec = 0;
   dayr = 0;
   dowr = 0;
   monthr = 0;
   yearr = 0;


#pragma asm
    MOV ILM, #7              ; allow all interrupts with level<7
    OR  CCR, #H'40    ; enable interrupts
#pragma endasm

    while(1);         /* dummy loop (wait for interrupt) */

}



/*---------------------- function bodies ----------------------*/



/*---------------------------------------------------------------------*/
/* Initial Routine for ICU : This routine sets the ICU0 as input */
/*---------------------------------------------------------------------*/

void Init_ICU(void)
{
    ICR07 = 6;                    /* INT level 6 for ICU interrupt */

    ICC_IRE0 = 0;                 /* 1=Interrupt enable for ICU#0..3 */
    ICC_IRE1 = 1;
    ICC_IRE2 = 0;
    ICC_IRE3 = 0;

    ICC_IR0  = 0;                 /* Interrupt request (set to 0) */
    ICC_IR1  = 0;
    ICC_IR2  = 0;
```

OBSOLETE

```
    ICC_IR3  = 0;

    ICC_EG0  = 0;                  /* detect : 0=no edges, 1=rising edges */
    ICC_EG1  = 1;                  /*          2=falling or 3=both edges */
    ICC_EG2  = 0;
    ICC_EG3  = 0;
}


/*---------------------------------------------------------------------*/
/* Interrupt Service Routine : Input capture unit 1 (Pin P65)    */
/*---------------------------------------------------------------------*/
__interrupt
void   ICU1(void)
{
    ICC_IR1 = 0;                          /* clear interrupt request */
    if (ICC_EG1 == 1)                            /* If rising edge was detected */
    {
      dist = ((ICR1 - t1) / 10000);              /* measure distance from last puls */
      ICC_EG1  = 2;                       /* detect falling edge now */
      TCCR_CLR = 0;                       /* reset 24-bit-timer */
    }
    else                                  /* If falling edge was detected */
    {
      t1 = ICR1;
      puls = (t1 / 1000);                 /* puls : 400..500=100ms ; 800..900=200ms */
      if ((puls > 400) && (puls < 500))   /* 0 received (100ms) */
      {
       if (dist < 700)                            /* within normal protocol ? */
       {
         sec++;                                   /* next second */
    if ((sec == 10) || (sec == 20) || (sec == 30) || (sec == 40) || (sec == 50))
          printdate(dowr,dayr,monthr,yearr);      /* show date every 10 seconds */
    else
          printtime(hrr,minr,sec);                /* show time */
        if (begin == 1)                   /* if tracing protocol */
        {
         index++;                         /* increase mark */
     if (index == 58)                     /* protocol finished ? */
         {
       hrr = hr;                          /* update time & date */
       minr = min;
       dayr = day;
       dowr = dow;
       monthr = month;
       yearr = year;
     }
         }
       }
       else                               /* begin of protocol and new minute */
       {
        begin = 1;
        index = 0;
        sec = 0;
        hr = 0;
        min = 0;
        day = 0;
        dow = 0;
        month = 0;
        year = 0;
       }
```

OBSOLETE

```
      }
      if ((puls > 800) && (puls < 900))  /* 1 received */
      {
       if (dist < 700)                          /* within normal protocol ? */
       {
         sec++;                                 /* next second */
    if ((sec == 10) || (sec == 20) || (sec == 30) || (sec == 40) || (sec == 50))
          printdate(dowr,dayr,monthr,yearr);   /* show date every 10 seconds */
    else
          printtime(hrr,minr,sec);              /* show time */
        if (begin == 1)                 /* tracing marks ? */
        {
         index++;                       /* increase mark */
         decode();                      /* decode protocol */
     if (index == 58)                   /* protocol finished ? */
         {
       hrr = hr;                        /* update time & date */
       minr = min;
       dayr = day;
       dowr = dow;
       monthr = month;
       yearr = year;
     }
        }
      }
       else
       {
    begin = 0;                          /* WRONG ! Do not begin to decode */
        sec = 0;
       }
      }
      ICC_EG1  = 1;  /* detect raising edge again */
    }
}

/*-------------------------------------------------------------*/
/* Decode : This procedure decodes the puls-code into time & date*/
/* currently no parity-check (marks 28,35,58), no additional     */
/* information (backup antenna, time zone etc)                   */
/*-------------------------------------------------------------*/

void decode(void)
{
  switch(index)                         /* index=second marks on "1" */
  {
   case 21: {min++; break;}                     /* 21-27 minutes */
   case 22: {min = min + 2; break;}
   case 23: {min = min + 4; break;}
   case 24: {min = min + 8; break;}
   case 25: {min = min + 10; break;}
   case 26: {min = min + 20; break;}
   case 27: {min = min + 40; break;}
   case 29: {hr++; break;}              /* 29-34 hours */
   case 30: {hr = hr + 2; break;}
   case 31: {hr = hr + 4; break;}
   case 32: {hr = hr + 8; break;}
   case 33: {hr = hr + 10; break;}
   case 34: {hr = hr + 40; break;}
   case 36: {day++; break;}                     /* 36-41 day in month */
   case 37: {day = day + 2; break;}
   case 38: {day = day + 4; break;}
```

OBSOLETE

```
    case 39: {day = day + 8; break;}
    case 40: {day = day + 10; break;}
    case 41: {day = day + 20; break;}
    case 42: {dow++; break;}                    /* 42-44 day in week */
    case 43: {dow = dow + 2; break;}
    case 44: {dow = dow + 4; break;}
    case 45: {month++; break;}                  /* 45-49 month */
    case 46: {month = month + 2; break;}
    case 47: {month = month + 4; break;}
    case 48: {month = month + 8; break;}
    case 49: {month = month + 10; break;}
    case 50: {year++; break;}                   /* 50-57 year */
    case 51: {year = year + 2; break;}
    case 52: {year = year + 4; break;}
    case 53: {year = year + 8; break;}
    case 54: {year = year + 10; break;}
    case 55: {year = year + 20; break;}
    case 56: {year = year + 40; break;}
    case 57: {year = year + 80; break;}
  }
}

/*-------------------------------------------------------------*/
void Init_Timer24(void)          /* init free run timer */
{
    ICR06 = 6;                   /* INT level 6 */

    TCCR_STP = 0;                /* stop timer */

    TCCR_CLR = 0;                /* 0=clear timer */
    TCCR_PR0 = 0;                /* 0=select clock : PLL/4 */
    TCCR_IVF = 0;                /* 0=clear interrupt request */
    TCCR_IVFE = 1;               /* 1=enable overflow INT */
    TCCR_TIM = 0;                /* 0=clear interrupt request */
    TCCR_TIME = 0;               /* 1=enable bit interm. INT */
    TCCR_TIS1 = 0;               /* 01=set bit period to 1024us */
    TCCR_TIS0 = 1;

/*  TCCR_STP = 1;                /* start timer (disabled) */
}

/*-------------------------------------------------------------*/
/* Interrupt Service Routine : 24-bit timer overflow           */
/* occurs every 4.1 seconds                                    */
/* currently only used to indicate that no signal is available */
/*-------------------------------------------------------------*/

__interrupt
void    Timer24of(void)
{
    TCCR_IVF = 0;                /* clear int request */
                    /* occurs every 4.194 seconds */

    print("NO DFC-SIGNAL...");   /* show error massage */

}



/*-------------------------------------------------------------*/
/* Initial Routine for 16-bit timer0 : This routine sets the   */
```

OBSOLETE

```
/* control register bits for reload-timer use and writes the     */
/* appropriate value to the reload-register dep. on clock speed  */
/*---------------------------------------------------------------*/

void Init_Timer16(void)
{
    ICR09 = 6;                      /* INT level 6 for timer interrupt */

         TMCSR0_CSL1 = 1;           /* Clock : 00 = PLL/2 (with 16MHz = 0.125us) */
    TMCSR0_CSL0 = 0;        /* 01 = PLL/8 (0.5us) ; 10 = PLL/32 (2us) */
    TMCSR0_MOD2 = 0;        /* 000 = Input trigger pin disabled */
    TMCSR0_MOD1 = 0;
    TMCSR0_MOD0 = 0;
    TMCSR0_OUTE = 0;        /* 0 = TOUT-pin disabled */
    TMCSR0_OUTL = 0;        /* 0 = TOUT-pin output level */
    TMCSR0_RELD = 1;        /* 1 = reload timer; 0 = one shot timer */
    TMCSR0_INTE = 1;        /* 1 = Interrupt enable */
    TMCSR0_UF   = 0;        /* 0 = Underflow Interrupt disable */
    TMCSR0_CNTE = 1;        /* 1 = timer enable - wait for trigger */

         TMRLR0 = 50000;            /* Reload-value for 16 MHz-clock source */

    TMCSR0_TRG = 1;         /* Trigger : ...Go ! */

}

/*---------------------------------------------------------------*/
/* Interrupt Service Routine : 16-bit timer 0                    */
/* occurs every second - counts seconds up when no signal is     */
/* available. Currently disabled...                              */
/*---------------------------------------------------------------*/
__interrupt
void Timer16(void)
{

 TMCSR0_UF = 0;                     /* clear int request */

 counter++;
 if (counter == 10)                 /* 10 * 100msec = 1s */
 {
   if (begin == 0) {sec++;};              /* increase second only if no signal */

   if (sec == 60)                   /* check beginning of new min,hour */
   {
    sec = 0;
    minr++;
    if (minr == 60)
    {
     minr = 0;
     hrr++;
     if (hrr == 24)
     {
      hrr = 0;
     }
    }
   }

            /* every 10 seconds display date for one second */

   if ((sec == 10) || (sec ==20) || (sec == 30) || (sec == 40) || (sec == 50))
   {
```

OBSOLETE

```
      printdate(dowr,dayr,monthr,yearr);
    }
    else
    {
      printtime(hrr,minr,sec);
    }
    counter = 0;
  }
}



/*-------------------------------------------------------------------*/
/* Display routines (Display LTN111 connected to port 6)             */
/*-------------------------------------------------------------------*/

void initdisp(void)         /* This Routine initializes the LCD on port6 */

{
    DDR6=0x0DF;             /* Port 6 Output except P65 (ICU1 !) */
    PDR6=0;                 /* Port 6 Off*/

    PDR6=0x013;             /* Startup sequence */
    PDR6=0x03;
      wait(1000);
    PDR6=0x013;
    PDR6=0x03;
      wait(1000);
    PDR6=0x013;
    PDR6=0x03;
      wait(1000);
    PDR6=0x012;
    PDR6=0x02;

    outb(0x028);            /* Switch to 4-bit mode */
    outb(0x0C);             /* Cursor Off (on=0x0F) */
    outb(0x06);             /* No shift */
    outb(0x03);             /* Cursor home */
    outb(0x01);             /* Display clear */
}



void outb(unsigned char a) /* send one byte to the display */
{
    BYTE b;

    cursor++;
    if (cursor == 9)
    {
     PDR6=0x01C;            /* correct position !LNT111-R only!*/
     PDR6=0x00C;
     PDR6=0x010;
     PDR6=0;
     busy();
    }
    b=(a & 0x0F0);          /* shift upper nibble */
    b = b >> 4;             /* to lower nibble */
```

OBSOLETE

```
    if (a & 0x080)  {b=(b | 0x080);};
              /* but keep Bit 7 */
    b=(b & 135);            /* set other bits to zero  */
    b=(b | 16);             /* set E line  */
    PDR6=b;                 /* send to LCD  */
    b=(b & 135);            /* clear E line  */
    PDR6=b;                 /* send to LCD  */
    b=(a & 143);            /* take lower nibble  */
    b=(b | 16);             /* set E line */
    PDR6=b;                 /* send to LCD  */
    b=(b & 143);            /* clear E line  */
    PDR6=b;                 /* send to LCD  */
    busy();                 /* wait for busy-line */

}

void busy(void)             /* This Routine polls the busy-line */
{
    BYTE b;
    PDR6=0;                 /* Port 3 Off before reading ! */
    b=1;
    while (b)           /* wait for Busy-line */
    {
     DDR6=0x0D0;            /* set Bus as input to read Bit .3 (Busy) */
     PDR6=0x05F;            /* busy request */
     b=PDR6_PD63;           /* read Port  */
     PDR6_PD64 = 0;
     PDR6_PD64 = 1;  /* toggle E */
     PDR6_PD64 = 0;
     PDR6_PD66 = 0;
     DDR6=0x0DF;            /* reset Port to output */
    }
}




void print(char *Name2)    /* This Routine displays a String */

{
 unsigned char c;
 BYTE b;
 int i,l;
 outb(1);           /* Display clear */
 l=strlen(Name2);
 cursor=0;
 for (i=0; i<l; i++)             /* go through string */
  {
   c=(Name2[i]);            /* pick char */
   b=(c | 128);
   outb(b);                 /* and display it */
  }
}


void printnum(int n)             /* show integer value on LCD display */
{
   float x;
   int l;
```

OBSOLETE

```
    if (n < 10)             /* only one digit value */
    {
     outb(0x0b0);
     outb(0x0b0);
     outb((n+48) | 128);
    }
    else if (n >= 10 && n<100)  /* two digit value */
    {
     outb(0x0b0);
     outb(((n/10)+48) | 128);
     x = n-(10*(n / 10));
     l = x;
     outb((l+48) | 128);
    }
    else if (n >= 100)          /* show three digits */
    {
     outb(((n/100)+48) | 128);
     x = n-(100*(n / 100));
     n = x;
     outb(((n/10)+48) | 128);
     x = n-(10*(n / 10));
     l = x;
     outb((l+48) | 128);
    }

}


void printdate(int dw,int dr,int mr, int yr)   /* show date on LCD display  */
{
    float x;
    int l;

    outb(1);                        /* Display clear */

    switch(dw)                      /* display day of week */
    {
     case 1:{outb(77 | 128); outb(79 | 128); break;}   /* MO */
     case 2:{outb(84 | 128); outb(85 | 128); break;}   /* TU */
     case 3:{outb(87 | 128); outb(69 | 128); break;}   /* WE */
     case 4:{outb(84 | 128); outb(72 | 128); break;}   /* TH */
     case 5:{outb(70 | 128); outb(82 | 128); break;}   /* FR */
     case 6:{outb(83 | 128); outb(65 | 128); break;}   /* SA */
     case 7:{outb(83 | 128); outb(85 | 128); break;}   /* SU */
    }
    outb(32 | 128);


    if (dr < 10)                    /* day in month : only one digit value */
    {
     outb(0x0b0);
     outb((dr+48) | 128);
    }
    else if (dr >= 10 && dr<100)   /* two digit value */
    {
     outb(((dr/10)+48) | 128);
     x = dr-(10*(dr / 10));
     l = x;
     outb((l+48) | 128);
    }
```

OBSOLETE

```
    outb(46 | 128);

    if (mr < 10)                    /* month : only one digit value */
    {
     outb(0x0b0);
     outb((mr+48) | 128);
    }
    else if (mr >= 10 && mr<100)    /* two digit value */
    {
     outb(((mr/10)+48) | 128);
     x = mr-(10*(mr / 10));
     l = x;
     outb((l+48) | 128);
    }

    outb(46 | 128);

    if (yr < 10)                    /* year : only one digit value */
    {
     outb(0x0b0);
     outb((yr+48) | 128);
    }
    else if (yr >= 10 && yr<100)    /* two digit value */
    {
     outb(((yr/10)+48) | 128);
     x = yr-(10*(yr / 10));
     l = x;
     outb((l+48) | 128);
    }
}


void printtime(int h,int m,int s) /* show time on LCD display hh:mm:ss */
{
    float x;
    int l;
    outb(1);                        /* Display clear */

    if (h < 10)                     /* hours : only one digit value */
    {
     outb(0x0b0);
     outb((h+48) | 128);
    }
    else if (h >= 10 && h<100)      /* two digit value */
    {
     outb(((h/10)+48) | 128);
     x = h-(10*(h / 10));
     l = x;
     outb((l+48) | 128);
    }

    outb(58 | 128);

    if (m < 10)                     /* minutes : only one digit value */
    {
     outb(0x0b0);
     outb((m+48) | 128);
    }
    else if (m >= 10 && m<100)      /* two digit value */
    {
     outb(((m/10)+48) | 128);
```

OBSOLETE

```
     x = m-(10*(m / 10));
     l = x;
     outb((l+48) | 128);
    }

    outb(58 | 128);

    if (s < 10)                    /* seconds : only one digit value */
    {
     outb(0x0b0);
     outb((s+48) | 128);
    }
    else if (s >= 10 && s<100)     /* two digit value */
    {
     outb(((s/10)+48) | 128);
     x = s-(10*(s / 10));
     l = x;
     outb((l+48) | 128);
    }
}

/*---------------------------------------------------------------*/
void wait(int i)
{
  for (; i ; i--);   /* very simple delay loop */
}
/*---------------------------------------------------------------*/
```

OBSOLETE

2 Document History

Document Title: AN204944 - DCF77-time signals with the MB90670/5 series by using the ICU

Document Number: 002-04944

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	IKHI	02/18/1997	Initial release
			06/28/2000	New format
*A	5097388	IKHI	02/26/2016	Migrated Spansion Application Note MCU-AN-390012-E-V11 to Cypress format This AN is for old product, So this should be obsoleted.

OBSOLETE

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

cypress.com/psoc

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums |Blogs | Video |Training

Technical Support

cypress.com/support

OBSOLETE

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 1997-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you under its copyright rights in the Software, a personal, non-exclusive, nontransferable license (without the right to sublicense) (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units. Cypress also grants you a personal, non-exclusive, nontransferable, license (without the right to sublicense) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely to the minimum extent that is necessary for you to exercise your rights under the copyright license granted in the previous sentence. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and Company shall and hereby does release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. Company shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.