Writing firmware for the EZ-USB

NOTE: This page is for GECKO3 firmware developers only and shows how the USB Controller EZ-USB can be programmed.

Installing the tools for Windows

Before you can write firmware for the EZ-USB FX2 chip (CY7C68013A from Cypress) you need to install the Cypress Development Kit for the manuals, examples and the Software to download your firmware to the EZ-USB.

1. Download and install the CY3684 EZ-USB FX2LP Development Kit from Cypress (`SETUP_FX2LP_DVK_xxxx.exe`). This Kit uses the Keil C compiler. Instead of it we will use the open source and platform independent SDCC (Small Device C Compiler).
2. Download and install SDCC for Windows (sdcc-win32) from SourceForge. SDCC is an open source retargettable, optimizing ANSI - C compiler. The current version targets the Intel 8051, Zilog Z80, Dallas 80C390, Motorola HC08 and Microchip PIC MCUs.

Libraries

The libraries from Cypress are only compatible with the Keil compiler. We will use SDCC instead and therefore other libraries and header are needed. These libraries have been written from the guys of GNURadio and since this is open source we will use them as well.

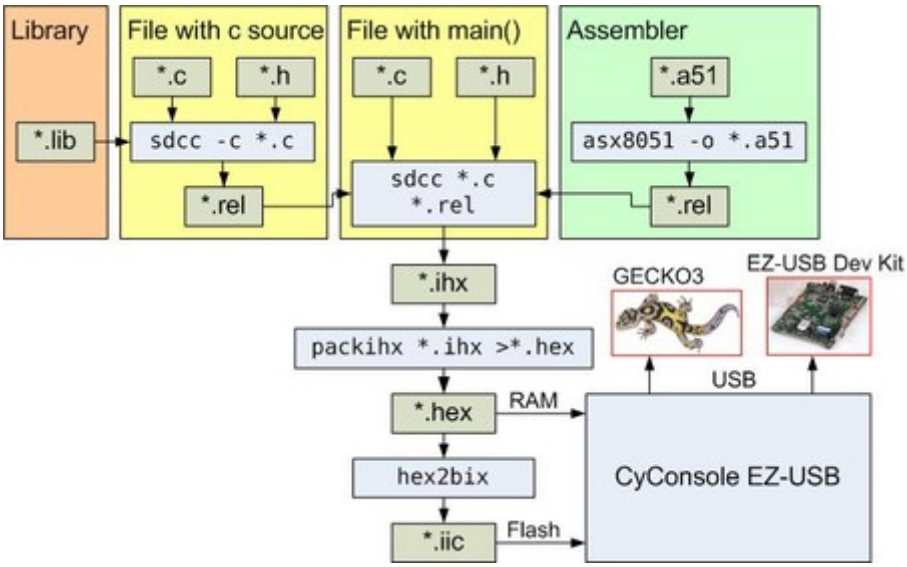
Compiling the firmware

You can use any text editor for creating and modifying the firmware. I used DEV C++ and Windows XP. To compile the firmware with SDCC I wrote the script `build.bat`. The firmware should be compilable under Linux as well. Under Windows it's recommended to work and compile from a local HDD.

build.bat

```
@echo off
cls

REM ## define the main file here: ##
set file=runninglight
REM ## define the include path (absolute) here: ##
set inc=C:\TEMP\runninglight\inc

echo ###############################################################################
echo # Firmware compiling Script for EZ-USB FX2, Copyright (C) 2008 by Lukas Kohler.#
echo # This program comes with ABSOLUTELY NO WARRANTY; for details see license.txt. #
echo # This is free software, and you are welcome to redistribute it under certain  #
echo # conditions; see license.txt for details.                                     #
echo ###############################################################################

REM #-------------------------------------------------------------------------
REM # File:     BUILD.BAT
REM # Contents: Batch file to build the EZ-USB FX2 firmware
REM # Date:     18.07.2008
REM #-------------------------------------------------------------------------
REM # Copyright 2008 by Lukas Kohler, Microlab BFH Biel
REM #-------------------------------------------------------------------------
REM # command line switch: -clean: delete temporary files
REM #-------------------------------------------------------------------------

echo ##                                                  Compiling libraries... ##
sdcc -c lib/i2c.c -I %inc%
```

```
IF ERRORLEVEL 1 goto ERRORLIB

echo ##                                                  Compiling C source file... ##
sdcc %file%.c i2c.rel -I %inc% -mmcs51 --xram-loc 0x2000 --xram-size 0x2000 --code-size 0x2000
IF ERRORLEVEL 1 goto ERRORMAIN

echo ##                                                          creating hex file... ##
packihx %file%.ihx >%file%.hex
IF ERRORLEVEL 1 goto ERRORHEX

echo ##                                                          creating bix file... ##
C:\cypress\usb\bin\hex2bix -i -r -m 0x16000 -v 0x04B4 -p 0x8613 -c 0x41 -f 0xC2 -o %file%.iic %file%.hex

echo ##                                              moving output files to res/... ##
move /Y *.hex res/
move /Y *.iic res/

echo ###############################################################################
echo ##                           finished wuthout Errors                           ##
echo ###############################################################################
goto END

:ERRORLIB
echo ## ERROR while compiling libraries! ##
goto END

:ERRORMAIN
echo ## ERROR while compiling main() ##
goto END

:ERRORMAIN
echo ## ERROR while compiling the HEX File ##

:END
REM ### usage: build -clean to remove intermediate files after build
if "%1" == "-clean" del *.lst
if "%1" == "-clean" del *.obj
if "%1" == "-clean" del *.m51
```



The batch file does this: From the libraries in the subfolder `lib/` (eg `i2c.c`) are `*.rel` files created and these compiled together with the file with the `main()` function. With the program program packihx is a `*.hex` file created. This `*.hex` file can be downloaded directly into the EZ-USB RAM without burning into the EEPROM. For burning the EEPROM the program `hex2bix` has to be used. This creates a `*.iic file`. The `*.hex` and `*.iic` files are then put into the `res/` subfolder.

```
wine Hex2bix.exe -I -F 0xC2 -C 0x41 -R -M 0xe200 -V 0x4705 -P 0x0002 -O test.iic usrp-dfu.hex
```

Simple Example

Lets say that you have three source files: foo1.c and foo2.c with some functions and foomain.c with the main() function. Compile first the fooX.c:

```
sdcc -c foo1.c
sdcc -c foo2.c
```

Then compile the source file containing the main() function and link the files together with the following command:

```
sdcc foomain.c foo1.rel foo2.rel
```

Alternatively, foomain.c can be separately compiled as well:

```
sdcc -c foomain.c
sdcc foomain.rel foo1.rel foo2.rel
```

Demo Firmware

Before starting with the GECKO3 firmware I created a simpler firmware. This example is a running light on Port PA and PB and sends data over the I2C interface. On the Cypress development board shows this a hex counter from 0..F on the 7segment display.

runninglight.c

```
/* Copyright (C) 2008 by Christoph Zimmermann and Lukas Kohler
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */

#define ALLOCATE_EXTERN
#include <fx2regs.h>
#include <i2c.h>

#define BLOCKSIZE 64

#define PORTOUTA    IOA
#define PORTOUTB    IOB
#define PORTOUTCFGA OEA
#define PORTOUTCFGB OEB
#define PORTIN      IOD
#define PORTINCFG   OED
#define PORTIND     PD0

#define ADD_SW      0x20
#define ADD_7SEG    0x21
```

```
/*****************************************************************************/
/*  \fn          void init_fx2(void)                                         */
/*****************************************************************************/
/*! \brief       Initializes the FX2 contoller
 *****************************************************************************
 *  \param[in]  -
 *  \param[out] -
 *  \return     -
 *****************************************************************************
 *  \author     Lukas Kohler
 *  \date       17.07.2007   LK created
 *  \test       OK
 *****************************************************************************/
void init_fx2(void)
{
  CPUCS = 0x08;       // 24MHz CPU-Clock
  EP1OUTBC = 0x01;    // Writing to EP1OUTBC rearms for an out Transfer
  PORTOUTCFGA = 0xff; // Port A (PA) is output
  PORTOUTCFGB = 0xff; // Port B (PB) is output
  PORTOUTA = 0x00;    // reset PA
  PORTOUTB = 0x00;    // reset PA
  PORTINCFG = 0xff;   // Eingabeport auf Ausgabe
  PORTIN = 0x00;      // Einmal auf Null stellen
  PORTINCFG = 0x00;   // Jetzt Eingabeport auf Eingabe einstellen
}

/*****************************************************************************/
/*  \fn          void main(void)                                             */
/*****************************************************************************/
/*! \brief       Main routine in endless loop. Displays a running light on
 *               Port PA and PB and sends data over the I2C interface.
 *               On the Cypress development board shows this I2C data a
 *               counter on the 7segment display. A pressed SW1 is reset
 *               of the counter
 *****************************************************************************
 *  \param[in]  -
 *  \param[out] -
 *  \return     -
 *****************************************************************************
 *  \author     Lukas Kohler
 *  \date       18.07.2007   LK created
 *  \test       OK
 *****************************************************************************/
void main(void)
{
  unsigned char cmd[1];
  unsigned char Digit[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x98, 0x88, 0x83, 0xc6,
0xa1, 0x86, 0x8e };
  // 7 segment data        0     1     2     3     4     5     6     7     8     9     A     B     C     D
E     F

  long x;
  unsigned char i;

  init_fx2(); // initialize FX2
  while(1)    // do endless
  {
    for(i=0; i<16; ) // Counter 0..15
    {
      switch(i)
```

```
      {
        case 0: PORTOUTB = 128, PORTOUTA = 0x00; break;
        case 1: PORTOUTB = 64; break;
        case 2: PORTOUTB = 32; break;
        case 3: PORTOUTB = 16; break;
        case 4: PORTOUTB = 8;  break;
        case 5: PORTOUTB = 4;  break;
        case 6: PORTOUTB = 2;  break;
        case 7: PORTOUTB = 1;  break;
        case 8: PORTOUTA = 1,  PORTOUTB = 0x00; break;
        case 9: PORTOUTA = 2;  break;
        case 10:PORTOUTA = 4;  break;
        case 11:PORTOUTA = 8;  break;
        case 12:PORTOUTA = 16; break;
        case 13:PORTOUTA = 32; break;
        case 14:PORTOUTA = 64; break;
        case 15:PORTOUTA = 128; break;
        default:PORTOUTA = 0xff,PORTOUTB = 0xff; // report errors
      }
      cmd[0] = Digit[i];
      i2c_write(ADD_7SEG, cmd, 1);   // write data to the 7segment controller

      i++;

      for(x=0; x<2000; x++){         // have a break - have a kitkat
        i2c_read (ADD_SW, cmd, 1);   // delay with reading switche SW1
        if(!(cmd[0]& 0x01)) i = 0;   // reset counter if SW1 is pressed
      }
    }
  }
}
```

/inc/i2c.h

```
/* -*- c++ -*- */
/*
 * Copyright 2003 Free Software Foundation, Inc.
 *
 * This file is part of GNU Radio
 *
 * GNU Radio is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3, or (at your option)
 * any later version.
 *
 * GNU Radio is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with GNU Radio; see the file COPYING.  If not, write to
 * the Free Software Foundation, Inc., 51 Franklin Street,
 * Boston, MA 02110-1301, USA.
 */

#ifndef _I2C_H_
#define _I2C_H_

// returns non-zero if successful, else 0
unsigned char i2c_read (unsigned char addr, unsigned char *buf, unsigned char len);
```

```
// returns non-zero if successful, else 0
unsigned char i2c_write (unsigned char addr, unsigned char *buf, unsigned char len);

#endif /* _I2C_H_ */
```

/inc/fx2regs.h

```
/* -*- c++ -*- */
/*
 * Copyright 2003 Free Software Foundation, Inc.
 *
 * This file is part of GNU Radio
 *
 * GNU Radio is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3, or (at your option)
 * any later version.
 *
 * GNU Radio is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with GNU Radio; see the file COPYING.  If not, write to
 * the Free Software Foundation, Inc., 51 Franklin Street,
 * Boston, MA 02110-1301, USA.
 */

/*
//-----------------------------------------------------------------------------
//   File:      FX2regs.h
//   Contents:   EZ-USB FX2 register declarations and bit mask definitions.
//
// $Archive: /USB/Target/Inc/fx2regs.h $
// $Date: 2007-07-20 20:44:38 -0700 (Fri, 20 Jul 2007) $
// $Revision: 6044 $
//
//
//   Copyright (c) 2000 Cypress Semiconductor, All rights reserved
//-----------------------------------------------------------------------------
*/


#ifndef FX2REGS_H   /* Header Sentry */
#define FX2REGS_H

#define ALLOCATE_EXTERN              // required for "right thing to happen" with fx2regs.h

/*
//-----------------------------------------------------------------------------
// FX2 Related Register Assignments
//-----------------------------------------------------------------------------

// The Ez-USB FX2 registers are defined here. We use FX2regs.h for register
// address allocation by using "#define ALLOCATE_EXTERN".
// When using "#define ALLOCATE_EXTERN", you get (for instance):
// xdata volatile BYTE OUT7BUF[64]   _at_   0x7B40;
// Such lines are created from FX2.h by using the preprocessor.
// Incidently, these lines will not generate any space in the resulting hex
```

```
// file; they just bind the symbols to the addresses for compilation.
// You just need to put "#define ALLOCATE_EXTERN" in your main program file;
// i.e. fw.c or a stand-alone C source file.
// Without "#define ALLOCATE_EXTERN", you just get the external reference:
// extern xdata volatile BYTE OUT7BUF[64]   ;//   0x7B40;
// This uses the concatenation operator "##" to insert a comment "//"
// to cut off the end of the line, "_at_   0x7B40;", which is not wanted.
*/


#ifdef ALLOCATE_EXTERN
#define EXTERN
#define _AT_(a) at a
#else
#define EXTERN extern
#define _AT_ ;// ## /
#endif

typedef unsigned char BYTE;
typedef unsigned short WORD;

EXTERN xdata _AT_(0xE400) volatile BYTE GPIF_WAVE_DATA[128];
EXTERN xdata _AT_(0xE480) volatile BYTE RES_WAVEDATA_END  ;

// General Configuration

EXTERN xdata _AT_(0xE600) volatile BYTE CPUCS             ;  // Control & Status
EXTERN xdata _AT_(0xE601) volatile BYTE IFCONFIG          ;  // Interface Configuration
EXTERN xdata _AT_(0xE602) volatile BYTE PINFLAGSAB        ;  // FIFO FLAGA and FLAGB Assignments
EXTERN xdata _AT_(0xE603) volatile BYTE PINFLAGSCD        ;  // FIFO FLAGC and FLAGD Assignments
EXTERN xdata _AT_(0xE604) volatile BYTE FIFORESET         ;  // Restore FIFOS to default state
EXTERN xdata _AT_(0xE605) volatile BYTE BREAKPT           ;  // Breakpoint
EXTERN xdata _AT_(0xE606) volatile BYTE BPADDRH           ;  // Breakpoint Address H
EXTERN xdata _AT_(0xE607) volatile BYTE BPADDRL           ;  // Breakpoint Address L
EXTERN xdata _AT_(0xE608) volatile BYTE UART230           ;  // 230 Kbaud clock for T0,T1,T2
EXTERN xdata _AT_(0xE609) volatile BYTE FIFOPINPOLAR      ;  // FIFO polarities
EXTERN xdata _AT_(0xE60A) volatile BYTE REVID             ;  // Chip Revision
EXTERN xdata _AT_(0xE60B) volatile BYTE REVCTL            ;  // Chip Revision Control

// Endpoint Configuration

EXTERN xdata _AT_(0xE610) volatile BYTE EP1OUTCFG         ;  // Endpoint 1-OUT Configuration
EXTERN xdata _AT_(0xE611) volatile BYTE EP1INCFG          ;  // Endpoint 1-IN Configuration
EXTERN xdata _AT_(0xE612) volatile BYTE EP2CFG            ;  // Endpoint 2 Configuration
EXTERN xdata _AT_(0xE613) volatile BYTE EP4CFG            ;  // Endpoint 4 Configuration
EXTERN xdata _AT_(0xE614) volatile BYTE EP6CFG            ;  // Endpoint 6 Configuration
EXTERN xdata _AT_(0xE615) volatile BYTE EP8CFG            ;  // Endpoint 8 Configuration
EXTERN xdata _AT_(0xE618) volatile BYTE EP2FIFOCFG        ;  // Endpoint 2 FIFO configuration
EXTERN xdata _AT_(0xE619) volatile BYTE EP4FIFOCFG        ;  // Endpoint 4 FIFO configuration
EXTERN xdata _AT_(0xE61A) volatile BYTE EP6FIFOCFG        ;  // Endpoint 6 FIFO configuration
EXTERN xdata _AT_(0xE61B) volatile BYTE EP8FIFOCFG        ;  // Endpoint 8 FIFO configuration
EXTERN xdata _AT_(0xE620) volatile BYTE EP2AUTOINLENH     ;  // Endpoint 2 Packet Length H (IN only)
EXTERN xdata _AT_(0xE621) volatile BYTE EP2AUTOINLENL     ;  // Endpoint 2 Packet Length L (IN only)
EXTERN xdata _AT_(0xE622) volatile BYTE EP4AUTOINLENH     ;  // Endpoint 4 Packet Length H (IN only)
EXTERN xdata _AT_(0xE623) volatile BYTE EP4AUTOINLENL     ;  // Endpoint 4 Packet Length L (IN only)
EXTERN xdata _AT_(0xE624) volatile BYTE EP6AUTOINLENH     ;  // Endpoint 6 Packet Length H (IN only)
EXTERN xdata _AT_(0xE625) volatile BYTE EP6AUTOINLENL     ;  // Endpoint 6 Packet Length L (IN only)
EXTERN xdata _AT_(0xE626) volatile BYTE EP8AUTOINLENH     ;  // Endpoint 8 Packet Length H (IN only)
EXTERN xdata _AT_(0xE627) volatile BYTE EP8AUTOINLENL     ;  // Endpoint 8 Packet Length L (IN only)
EXTERN xdata _AT_(0xE630) volatile BYTE EP2FIFOPFH        ;  // EP2 Programmable Flag trigger H
EXTERN xdata _AT_(0xE631) volatile BYTE EP2FIFOPFL        ;  // EP2 Programmable Flag trigger L
```

```
EXTERN xdata _AT_(0xE632) volatile BYTE EP4FIFOPFH        ;  // EP4 Programmable Flag trigger H
EXTERN xdata _AT_(0xE633) volatile BYTE EP4FIFOPFL        ;  // EP4 Programmable Flag trigger L
EXTERN xdata _AT_(0xE634) volatile BYTE EP6FIFOPFH        ;  // EP6 Programmable Flag trigger H
EXTERN xdata _AT_(0xE635) volatile BYTE EP6FIFOPFL        ;  // EP6 Programmable Flag trigger L
EXTERN xdata _AT_(0xE636) volatile BYTE EP8FIFOPFH        ;  // EP8 Programmable Flag trigger H
EXTERN xdata _AT_(0xE637) volatile BYTE EP8FIFOPFL        ;  // EP8 Programmable Flag trigger L
EXTERN xdata _AT_(0xE640) volatile BYTE EP2ISOINPKTS      ;  // EP2 (if ISO) IN Packets per frame (1-3)
EXTERN xdata _AT_(0xE641) volatile BYTE EP4ISOINPKTS      ;  // EP4 (if ISO) IN Packets per frame (1-3)
EXTERN xdata _AT_(0xE642) volatile BYTE EP6ISOINPKTS      ;  // EP6 (if ISO) IN Packets per frame (1-3)
EXTERN xdata _AT_(0xE643) volatile BYTE EP8ISOINPKTS      ;  // EP8 (if ISO) IN Packets per frame (1-3)
EXTERN xdata _AT_(0xE648) volatile BYTE INPKTEND          ;  // Force IN Packet End
EXTERN xdata _AT_(0xE649) volatile BYTE OUTPKTEND         ;  // Force OUT Packet End

// Interrupts

EXTERN xdata _AT_(0xE650) volatile BYTE EP2FIFOIE         ;  // Endpoint 2 Flag Interrupt Enable
EXTERN xdata _AT_(0xE651) volatile BYTE EP2FIFOIRQ        ;  // Endpoint 2 Flag Interrupt Request
EXTERN xdata _AT_(0xE652) volatile BYTE EP4FIFOIE         ;  // Endpoint 4 Flag Interrupt Enable
EXTERN xdata _AT_(0xE653) volatile BYTE EP4FIFOIRQ        ;  // Endpoint 4 Flag Interrupt Request
EXTERN xdata _AT_(0xE654) volatile BYTE EP6FIFOIE         ;  // Endpoint 6 Flag Interrupt Enable
EXTERN xdata _AT_(0xE655) volatile BYTE EP6FIFOIRQ        ;  // Endpoint 6 Flag Interrupt Request
EXTERN xdata _AT_(0xE656) volatile BYTE EP8FIFOIE         ;  // Endpoint 8 Flag Interrupt Enable
EXTERN xdata _AT_(0xE657) volatile BYTE EP8FIFOIRQ        ;  // Endpoint 8 Flag Interrupt Request
EXTERN xdata _AT_(0xE658) volatile BYTE IBNIE             ;  // IN-BULK-NAK Interrupt Enable
EXTERN xdata _AT_(0xE659) volatile BYTE IBNIRQ            ;  // IN-BULK-NAK interrupt Request
EXTERN xdata _AT_(0xE65A) volatile BYTE NAKIE             ;  // Endpoint Ping NAK interrupt Enable
EXTERN xdata _AT_(0xE65B) volatile BYTE NAKIRQ            ;  // Endpoint Ping NAK interrupt Request
EXTERN xdata _AT_(0xE65C) volatile BYTE USBIE             ;  // USB Int Enables
EXTERN xdata _AT_(0xE65D) volatile BYTE USBIRQ            ;  // USB Interrupt Requests
EXTERN xdata _AT_(0xE65E) volatile BYTE EPIE              ;  // Endpoint Interrupt Enables
EXTERN xdata _AT_(0xE65F) volatile BYTE EPIRQ             ;  // Endpoint Interrupt Requests
EXTERN xdata _AT_(0xE660) volatile BYTE GPIFIE            ;  // GPIF Interrupt Enable
EXTERN xdata _AT_(0xE661) volatile BYTE GPIFIRQ           ;  // GPIF Interrupt Request
EXTERN xdata _AT_(0xE662) volatile BYTE USBERRIE          ;  // USB Error Interrupt Enables
EXTERN xdata _AT_(0xE663) volatile BYTE USBERRIRQ         ;  // USB Error Interrupt Requests
EXTERN xdata _AT_(0xE664) volatile BYTE ERRCNTLIM         ;  // USB Error counter and limit
EXTERN xdata _AT_(0xE665) volatile BYTE CLRERRCNT         ;  // Clear Error Counter EC[3..0]
EXTERN xdata _AT_(0xE666) volatile BYTE INT2IVEC          ;  // Interupt 2 (USB) Autovector
EXTERN xdata _AT_(0xE667) volatile BYTE INT4IVEC          ;  // Interupt 4 (FIFOS & GPIF) Autovector
EXTERN xdata _AT_(0xE668) volatile BYTE INTSETUP          ;  // Interrupt 2&4 Setup

// Input/Output

EXTERN xdata _AT_(0xE670) volatile BYTE PORTACFG          ;  // I/O PORTA Alternate Configuration
EXTERN xdata _AT_(0xE671) volatile BYTE PORTCCFG          ;  // I/O PORTC Alternate Configuration
EXTERN xdata _AT_(0xE672) volatile BYTE PORTECFG          ;  // I/O PORTE Alternate Configuration
EXTERN xdata _AT_(0xE678) volatile BYTE I2CS              ;  // Control & Status
EXTERN xdata _AT_(0xE679) volatile BYTE I2DAT             ;  // Data
EXTERN xdata _AT_(0xE67A) volatile BYTE I2CTL             ;  // I2C Control
EXTERN xdata _AT_(0xE67B) volatile BYTE XAUTODAT1         ;  // Autoptr1 MOVX access
EXTERN xdata _AT_(0xE67C) volatile BYTE XAUTODAT2         ;  // Autoptr2 MOVX access

#define EXTAUTODAT1 XAUTODAT1
#define EXTAUTODAT2 XAUTODAT2

// USB Control

EXTERN xdata _AT_(0xE680) volatile BYTE USBCS             ;  // USB Control & Status
EXTERN xdata _AT_(0xE681) volatile BYTE SUSPEND           ;  // Put chip into suspend
EXTERN xdata _AT_(0xE682) volatile BYTE WAKEUPCS          ;  // Wakeup source and polarity
EXTERN xdata _AT_(0xE683) volatile BYTE TOGCTL            ;  // Toggle Control
```

```
EXTERN xdata _AT_(0xE684) volatile BYTE USBFRAMEH         ;  // USB Frame count H
EXTERN xdata _AT_(0xE685) volatile BYTE USBFRAMEL         ;  // USB Frame count L
EXTERN xdata _AT_(0xE686) volatile BYTE MICROFRAME        ;  // Microframe count, 0-7
EXTERN xdata _AT_(0xE687) volatile BYTE FNADDR            ;  // USB Function address

// Endpoints

EXTERN xdata _AT_(0xE68A) volatile BYTE EP0BCH            ;  // Endpoint 0 Byte Count H
EXTERN xdata _AT_(0xE68B) volatile BYTE EP0BCL            ;  // Endpoint 0 Byte Count L
EXTERN xdata _AT_(0xE68D) volatile BYTE EP1OUTBC          ;  // Endpoint 1 OUT Byte Count
EXTERN xdata _AT_(0xE68F) volatile BYTE EP1INBC           ;  // Endpoint 1 IN Byte Count
EXTERN xdata _AT_(0xE690) volatile BYTE EP2BCH            ;  // Endpoint 2 Byte Count H
EXTERN xdata _AT_(0xE691) volatile BYTE EP2BCL            ;  // Endpoint 2 Byte Count L
EXTERN xdata _AT_(0xE694) volatile BYTE EP4BCH            ;  // Endpoint 4 Byte Count H
EXTERN xdata _AT_(0xE695) volatile BYTE EP4BCL            ;  // Endpoint 4 Byte Count L
EXTERN xdata _AT_(0xE698) volatile BYTE EP6BCH            ;  // Endpoint 6 Byte Count H
EXTERN xdata _AT_(0xE699) volatile BYTE EP6BCL            ;  // Endpoint 6 Byte Count L
EXTERN xdata _AT_(0xE69C) volatile BYTE EP8BCH            ;  // Endpoint 8 Byte Count H
EXTERN xdata _AT_(0xE69D) volatile BYTE EP8BCL            ;  // Endpoint 8 Byte Count L
EXTERN xdata _AT_(0xE6A0) volatile BYTE EP0CS             ;  // Endpoint  Control and Status
EXTERN xdata _AT_(0xE6A1) volatile BYTE EP1OUTCS          ;  // Endpoint 1 OUT Control and Status
EXTERN xdata _AT_(0xE6A2) volatile BYTE EP1INCS           ;  // Endpoint 1 IN Control and Status
EXTERN xdata _AT_(0xE6A3) volatile BYTE EP2CS             ;  // Endpoint 2 Control and Status
EXTERN xdata _AT_(0xE6A4) volatile BYTE EP4CS             ;  // Endpoint 4 Control and Status
EXTERN xdata _AT_(0xE6A5) volatile BYTE EP6CS             ;  // Endpoint 6 Control and Status
EXTERN xdata _AT_(0xE6A6) volatile BYTE EP8CS             ;  // Endpoint 8 Control and Status
EXTERN xdata _AT_(0xE6A7) volatile BYTE EP2FIFOFLGS       ;  // Endpoint 2 Flags
EXTERN xdata _AT_(0xE6A8) volatile BYTE EP4FIFOFLGS       ;  // Endpoint 4 Flags
EXTERN xdata _AT_(0xE6A9) volatile BYTE EP6FIFOFLGS       ;  // Endpoint 6 Flags
EXTERN xdata _AT_(0xE6AA) volatile BYTE EP8FIFOFLGS       ;  // Endpoint 8 Flags
EXTERN xdata _AT_(0xE6AB) volatile BYTE EP2FIFOBCH        ;  // EP2 FIFO total byte count H
EXTERN xdata _AT_(0xE6AC) volatile BYTE EP2FIFOBCL        ;  // EP2 FIFO total byte count L
EXTERN xdata _AT_(0xE6AD) volatile BYTE EP4FIFOBCH        ;  // EP4 FIFO total byte count H
EXTERN xdata _AT_(0xE6AE) volatile BYTE EP4FIFOBCL        ;  // EP4 FIFO total byte count L
EXTERN xdata _AT_(0xE6AF) volatile BYTE EP6FIFOBCH        ;  // EP6 FIFO total byte count H
EXTERN xdata _AT_(0xE6B0) volatile BYTE EP6FIFOBCL        ;  // EP6 FIFO total byte count L
EXTERN xdata _AT_(0xE6B1) volatile BYTE EP8FIFOBCH        ;  // EP8 FIFO total byte count H
EXTERN xdata _AT_(0xE6B2) volatile BYTE EP8FIFOBCL        ;  // EP8 FIFO total byte count L
EXTERN xdata _AT_(0xE6B3) volatile BYTE SUDPTRH           ;  // Setup Data Pointer high address byte
EXTERN xdata _AT_(0xE6B4) volatile BYTE SUDPTRL           ;  // Setup Data Pointer low address byte
EXTERN xdata _AT_(0xE6B5) volatile BYTE SUDPTRCTL         ;  // Setup Data Pointer Auto Mode
EXTERN xdata _AT_(0xE6B8) volatile BYTE SETUPDAT[8]       ;  // 8 bytes of SETUP data

// GPIF

EXTERN xdata _AT_(0xE6C0) volatile BYTE GPIFWFSELECT      ;  // Waveform Selector
EXTERN xdata _AT_(0xE6C1) volatile BYTE GPIFIDLECS        ;  // GPIF Done, GPIF IDLE drive mode
EXTERN xdata _AT_(0xE6C2) volatile BYTE GPIFIDLECTL       ;  // Inactive Bus, CTL states
EXTERN xdata _AT_(0xE6C3) volatile BYTE GPIFCTLCFG        ;  // CTL OUT pin drive
EXTERN xdata _AT_(0xE6C4) volatile BYTE GPIFADRH          ;  // GPIF Address H
EXTERN xdata _AT_(0xE6C5) volatile BYTE GPIFADRL          ;  // GPIF Address L

EXTERN xdata _AT_(0xE6CE) volatile BYTE GPIFTCB3          ;  // GPIF Transaction Count Byte 3
EXTERN xdata _AT_(0xE6CF) volatile BYTE GPIFTCB2          ;  // GPIF Transaction Count Byte 2
EXTERN xdata _AT_(0xE6D0) volatile BYTE GPIFTCB1          ;  // GPIF Transaction Count Byte 1
EXTERN xdata _AT_(0xE6D1) volatile BYTE GPIFTCB0          ;  // GPIF Transaction Count Byte 0

#define EP2GPIFTCH GPIFTCB1   // these are here for backwards compatibility
#define EP2GPIFTCL GPIFTCB0   // before REVE silicon (ie. REVB and REVD)
#define EP4GPIFTCH GPIFTCB1   // these are here for backwards compatibility
#define EP4GPIFTCL GPIFTCB0   // before REVE silicon (ie. REVB and REVD)
```

```
#define EP6GPIFTCH GPIFTCB1   // these are here for backwards compatibility
#define EP6GPIFTCL GPIFTCB0   // before REVE silicon (ie. REVB and REVD)
#define EP8GPIFTCH GPIFTCB1   // these are here for backwards compatibility
#define EP8GPIFTCL GPIFTCB0   // before REVE silicon (ie. REVB and REVD)

// EXTERN xdata volatile BYTE EP2GPIFTCH     _AT_ 0xE6D0;  // EP2 GPIF Transaction Count High
// EXTERN xdata volatile BYTE EP2GPIFTCL     _AT_ 0xE6D1;  // EP2 GPIF Transaction Count Low
EXTERN xdata _AT_(0xE6D2) volatile BYTE EP2GPIFFLGSEL     ;  // EP2 GPIF Flag select
EXTERN xdata _AT_(0xE6D3) volatile BYTE EP2GPIFPFSTOP     ;  // Stop GPIF EP2 transaction on prog. flag
EXTERN xdata _AT_(0xE6D4) volatile BYTE EP2GPIFTRIG       ;  // EP2 FIFO Trigger
// EXTERN xdata volatile BYTE EP4GPIFTCH     _AT_ 0xE6D8;  // EP4 GPIF Transaction Count High
// EXTERN xdata volatile BYTE EP4GPIFTCL     _AT_ 0xE6D9;  // EP4 GPIF Transactionr Count Low
EXTERN xdata _AT_(0xE6DA) volatile BYTE EP4GPIFFLGSEL     ;  // EP4 GPIF Flag select
EXTERN xdata _AT_(0xE6DB) volatile BYTE EP4GPIFPFSTOP     ;  // Stop GPIF EP4 transaction on prog. flag
EXTERN xdata _AT_(0xE6DC) volatile BYTE EP4GPIFTRIG       ;  // EP4 FIFO Trigger
// EXTERN xdata volatile BYTE EP6GPIFTCH     _AT_ 0xE6E0;  // EP6 GPIF Transaction Count High
// EXTERN xdata volatile BYTE EP6GPIFTCL     _AT_ 0xE6E1;  // EP6 GPIF Transaction Count Low
EXTERN xdata _AT_(0xE6E2) volatile BYTE EP6GPIFFLGSEL     ;  // EP6 GPIF Flag select
EXTERN xdata _AT_(0xE6E3) volatile BYTE EP6GPIFPFSTOP     ;  // Stop GPIF EP6 transaction on prog. flag
EXTERN xdata _AT_(0xE6E4) volatile BYTE EP6GPIFTRIG       ;  // EP6 FIFO Trigger
// EXTERN xdata volatile BYTE EP8GPIFTCH     _AT_ 0xE6E8;  // EP8 GPIF Transaction Count High
// EXTERN xdata volatile BYTE EP8GPIFTCL     _AT_ 0xE6E9;  // EP8GPIF Transaction Count Low
EXTERN xdata _AT_(0xE6EA) volatile BYTE EP8GPIFFLGSEL     ;  // EP8 GPIF Flag select
EXTERN xdata _AT_(0xE6EB) volatile BYTE EP8GPIFPFSTOP     ;  // Stop GPIF EP8 transaction on prog. flag
EXTERN xdata _AT_(0xE6EC) volatile BYTE EP8GPIFTRIG       ;  // EP8 FIFO Trigger
EXTERN xdata _AT_(0xE6F0) volatile BYTE XGPIFSGLDATH      ;  // GPIF Data H (16-bit mode only)
EXTERN xdata _AT_(0xE6F1) volatile BYTE XGPIFSGLDATLX     ;  // Read/Write GPIF Data L & trigger transac
EXTERN xdata _AT_(0xE6F2) volatile BYTE XGPIFSGLDATLNOX   ;  // Read GPIF Data L, no transac trigger
EXTERN xdata _AT_(0xE6F3) volatile BYTE GPIFREADYCFG      ;  // Internal RDY,Sync/Async, RDY5CFG
EXTERN xdata _AT_(0xE6F4) volatile BYTE GPIFREADYSTAT     ;  // RDY pin states
EXTERN xdata _AT_(0xE6F5) volatile BYTE GPIFABORT         ;  // Abort GPIF cycles

// UDMA

EXTERN xdata _AT_(0xE6C6) volatile BYTE FLOWSTATE         ; //Defines GPIF flow state
EXTERN xdata _AT_(0xE6C7) volatile BYTE FLOWLOGIC         ; //Defines flow/hold decision criteria
EXTERN xdata _AT_(0xE6C8) volatile BYTE FLOWEQ0CTL        ; //CTL states during active flow state
EXTERN xdata _AT_(0xE6C9) volatile BYTE FLOWEQ1CTL        ; //CTL states during hold flow state
EXTERN xdata _AT_(0xE6CA) volatile BYTE FLOWHOLDOFF       ;
EXTERN xdata _AT_(0xE6CB) volatile BYTE FLOWSTB           ; //CTL/RDY Signal to use as master data strobe
EXTERN xdata _AT_(0xE6CC) volatile BYTE FLOWSTBEDGE       ; //Defines active master strobe edge
EXTERN xdata _AT_(0xE6CD) volatile BYTE FLOWSTBHPERIOD    ; //Half Period of output master strobe
EXTERN xdata _AT_(0xE60C) volatile BYTE GPIFHOLDAMOUNT    ; //Data delay shift
EXTERN xdata _AT_(0xE67D) volatile BYTE UDMACRCH           ; //CRC Upper byte
EXTERN xdata _AT_(0xE67E) volatile BYTE UDMACRCL           ; //CRC Lower byte
EXTERN xdata _AT_(0xE67F) volatile BYTE UDMACRCQUAL       ; //UDMA In only, host terminated use only


// Debug/Test

EXTERN xdata _AT_(0xE6F8) volatile BYTE DBUG              ;  // Debug
EXTERN xdata _AT_(0xE6F9) volatile BYTE TESTCFG           ;  // Test configuration
EXTERN xdata _AT_(0xE6FA) volatile BYTE USBTEST           ;  // USB Test Modes
EXTERN xdata _AT_(0xE6FB) volatile BYTE CT1               ;  // Chirp Test--Override
EXTERN xdata _AT_(0xE6FC) volatile BYTE CT2               ;  // Chirp Test--FSM
EXTERN xdata _AT_(0xE6FD) volatile BYTE CT3               ;  // Chirp Test--Control Signals
EXTERN xdata _AT_(0xE6FE) volatile BYTE CT4               ;  // Chirp Test--Inputs

// Endpoint Buffers

EXTERN xdata _AT_(0xE740) volatile BYTE EP0BUF[64]        ;  // EP0 IN-OUT buffer
```

```
EXTERN xdata _AT_(0xE780) volatile BYTE EP1OUTBUF[64]     ;  // EP1-OUT buffer
EXTERN xdata _AT_(0xE7C0) volatile BYTE EP1INBUF[64]      ;  // EP1-IN buffer
EXTERN xdata _AT_(0xF000) volatile BYTE EP2FIFOBUF[1024]  ;  // 512/1024-byte EP2 buffer (IN or OUT)
EXTERN xdata _AT_(0xF400) volatile BYTE EP4FIFOBUF[1024]  ;  // 512 byte EP4 buffer (IN or OUT)
EXTERN xdata _AT_(0xF800) volatile BYTE EP6FIFOBUF[1024]  ;  // 512/1024-byte EP6 buffer (IN or OUT)
EXTERN xdata _AT_(0xFC00) volatile BYTE EP8FIFOBUF[1024]  ;  // 512 byte EP8 buffer (IN or OUT)

#undef EXTERN
#undef _AT_

/*-----------------------------------------------------------------------------
   Special Function Registers (SFRs)
   The byte registers and bits defined in the following list are based
   on the Synopsis definition of the 8051 Special Function Registers for EZ-USB.
    If you modify the register definitions below, please regenerate the file
    "ezregs.inc" which includes the same basic information for assembly inclusion.
-----------------------------------------------------------------------------*/

sfr at 0x80 IOA;
sfr at 0x81 SP;
sfr at 0x82 DPL;
sfr at 0x83 DPH;
sfr at 0x84 DPL1;
sfr at 0x85 DPH1;
sfr at 0x86 DPS;
         /*  DPS  */
         sbit at 0x86+0 SEL;
sfr at 0x87 PCON;   /*  PCON  */
         //sbit IDLE   = 0x87+0;
         //sbit STOP   = 0x87+1;
         //sbit GF0    = 0x87+2;
         //sbit GF1    = 0x87+3;
         //sbit SMOD0  = 0x87+7;
sfr at 0x88 TCON;
         /*  TCON  */
         sbit at 0x88+0 IT0;
         sbit at 0x88+1 IE0;
         sbit at 0x88+2 IT1;
         sbit at 0x88+3 IE1;
         sbit at 0x88+4 TR0;
         sbit at 0x88+5 TF0;
         sbit at 0x88+6 TR1;
         sbit at 0x88+7 TF1;
sfr at 0x89 TMOD;
         /*  TMOD  */
         //sbit M00    = 0x89+0;
         //sbit M10    = 0x89+1;
         //sbit CT0    = 0x89+2;
         //sbit GATE0  = 0x89+3;
         //sbit M01    = 0x89+4;
         //sbit M11    = 0x89+5;
         //sbit CT1    = 0x89+6;
         //sbit GATE1  = 0x89+7;
sfr at 0x8A TL0;
sfr at 0x8B TL1;
sfr at 0x8C TH0;
sfr at 0x8D TH1;
sfr at 0x8E CKCON;
         /*  CKCON  */
         //sbit MD0    = 0x89+0;
         //sbit MD1    = 0x89+1;
```

```
         //sbit MD2    = 0x89+2;
         //sbit T0M    = 0x89+3;
         //sbit T1M    = 0x89+4;
         //sbit T2M    = 0x89+5;
// sfr at 0x8F SPC_FNC; // Was WRS in Reg320
         /*  CKCON  */
         //sbit WRS    = 0x8F+0;
sfr at 0x90 IOB;
sfr at 0x91 EXIF; // EXIF Bit Values differ from Reg320
         /*  EXIF  */
         //sbit USBINT = 0x91+4;
         //sbit I2CINT = 0x91+5;
         //sbit IE4    = 0x91+6;
         //sbit IE5    = 0x91+7;
sfr at 0x92 MPAGE;
sfr at 0x98 SCON0;
         /*  SCON0  */
         sbit at 0x98+0 RI;
         sbit at 0x98+1 TI;
         sbit at 0x98+2 RB8;
         sbit at 0x98+3 TB8;
         sbit at 0x98+4 REN;
         sbit at 0x98+5 SM2;
         sbit at 0x98+6 SM1;
         sbit at 0x98+7 SM0;
sfr at 0x99 SBUF0;

sfr at 0x9A APTR1H;
sfr at 0x9B APTR1L;
sfr at 0x9C AUTODAT1;
sfr at 0x9D AUTOPTRH2;
sfr at 0x9E AUTOPTRL2;
sfr at 0x9F AUTODAT2;
sfr at 0xA0 IOC;
sfr at 0xA1 INT2CLR;
sfr at 0xA2 INT4CLR;

#define       AUTOPTRH1        APTR1H
#define       AUTOPTRL1        APTR1L

sfr at 0xA8 IE;
         /*  IE  */
         sbit at 0xA8+0 EX0;
         sbit at 0xA8+1 ET0;
         sbit at 0xA8+2 EX1;
         sbit at 0xA8+3 ET1;
         sbit at 0xA8+4 ES0;
         sbit at 0xA8+5 ET2;
         sbit at 0xA8+6 ES1;
         sbit at 0xA8+7 EA;

sfr at 0xAA EP2468STAT;
         /* EP2468STAT */
         //sbit EP2E   = 0xAA+0;
         //sbit EP2F   = 0xAA+1;
         //sbit EP4E   = 0xAA+2;
         //sbit EP4F   = 0xAA+3;
         //sbit EP6E   = 0xAA+4;
         //sbit EP6F   = 0xAA+5;
         //sbit EP8E   = 0xAA+6;
         //sbit EP8F   = 0xAA+7;
```

```
sfr at 0xAB EP24FIFOFLGS;
sfr at 0xAC EP68FIFOFLGS;
sfr at 0xAF AUTOPTRSETUP;
            /* AUTOPTRSETUP */
            // sbit EXTACC  = 0xAF+0;
            // sbit APTR1FZ = 0xAF+1;
            // sbit APTR2FZ = 0xAF+2;

sfr at 0xB0 IOD;
sfr at 0xB1 IOE;
sfr at 0xB2 OEA;
sfr at 0xB3 OEB;
sfr at 0xB4 OEC;
sfr at 0xB5 OED;
sfr at 0xB6 OEE;

sfr at 0xB8 IP;
         /*  IP  */
         sbit at 0xB8+0 PX0;
         sbit at 0xB8+1 PT0;
         sbit at 0xB8+2 PX1;
         sbit at 0xB8+3 PT1;
         sbit at 0xB8+4 PS0;
         sbit at 0xB8+5 PT2;
         sbit at 0xB8+6 PS1;

sfr at 0xBA EP01STAT;
sfr at 0xBB GPIFTRIG;

sfr at 0xBD GPIFSGLDATH;
sfr at 0xBE GPIFSGLDATLX;
sfr at 0xBF GPIFSGLDATLNOX;

sfr at 0xC0 SCON1;
         /*  SCON1  */
         sbit at 0xC0+0 RI1;
         sbit at 0xC0+1 TI1;
         sbit at 0xC0+2 RB81;
         sbit at 0xC0+3 TB81;
         sbit at 0xC0+4 REN1;
         sbit at 0xC0+5 SM21;
         sbit at 0xC0+6 SM11;
         sbit at 0xC0+7 SM01;
sfr at 0xC1 SBUF1;
sfr at 0xC8 T2CON;
         /*  T2CON  */
         sbit at 0xC8+0 CP_RL2;
         sbit at 0xC8+1 C_T2;
         sbit at 0xC8+2 TR2;
         sbit at 0xC8+3 EXEN2;
         sbit at 0xC8+4 TCLK;
         sbit at 0xC8+5 RCLK;
         sbit at 0xC8+6 EXF2;
         sbit at 0xC8+7 TF2;
sfr at 0xCA RCAP2L;
sfr at 0xCB RCAP2H;
sfr at 0xCC TL2;
sfr at 0xCD TH2;
sfr at 0xD0 PSW;
         /*  PSW  */
```

```
         sbit at 0xD0+0 P;
         sbit at 0xD0+1 FL;
         sbit at 0xD0+2 OV;
         sbit at 0xD0+3 RS0;
         sbit at 0xD0+4 RS1;
         sbit at 0xD0+5 F0;
         sbit at 0xD0+6 AC;
         sbit at 0xD0+7 CY;
sfr at 0xD8 EICON; // Was WDCON in DS80C320 EICON; Bit Values differ from Reg320
         /*  EICON  */
         sbit at 0xD8+3 INT6;
         sbit at 0xD8+4 RESI;
         sbit at 0xD8+5 ERESI;
         sbit at 0xD8+7 SMOD1;
sfr at 0xE0 ACC;
sfr at 0xE8 EIE; // EIE Bit Values differ from Reg320
                         /*  EIE  */
         sbit at 0xE8+0 EIUSB;
         sbit at 0xE8+1 EI2C;
         sbit at 0xE8+2 EIEX4;
         sbit at 0xE8+3 EIEX5;
         sbit at 0xE8+4 EIEX6;
sfr at 0xF0 B;
sfr at 0xF8 EIP; // EIP Bit Values differ from Reg320
                         /*  EIP  */
         sbit at 0xF8+0 PUSB;
         sbit at 0xF8+1 PI2C;
         sbit at 0xF8+2 EIPX4;
         sbit at 0xF8+3 EIPX5;
         sbit at 0xF8+4 EIPX6;

/*-----------------------------------------------------------------------------
   Bit Masks
-----------------------------------------------------------------------------*/

#define bmBIT0        1
#define bmBIT1        2
#define bmBIT2        4
#define bmBIT3        8
#define bmBIT4        16
#define bmBIT5        32
#define bmBIT6        64
#define bmBIT7        128

/* CPU Control & Status Register (CPUCS) */
#define bmPRTCSTB    bmBIT5
#define bmCLKSPD     (bmBIT4 | bmBIT3)
#define bmCLKSPD1    bmBIT4
#define bmCLKSPD0    bmBIT3
#define bmCLKINV     bmBIT2
#define bmCLKOE      bmBIT1
#define bm8051RES    bmBIT0
/* Port Alternate Configuration Registers */
/* Port A (PORTACFG) */
#define bmFLAGD      bmBIT7
#define bmINT1       bmBIT1
#define bmINT0       bmBIT0
/* Port C (PORTCCFG) */
#define bmGPIFA7     bmBIT7
#define bmGPIFA6     bmBIT6
#define bmGPIFA5     bmBIT5
```

```
#define bmGPIFA4     bmBIT4
#define bmGPIFA3     bmBIT3
#define bmGPIFA2     bmBIT2
#define bmGPIFA1     bmBIT1
#define bmGPIFA0     bmBIT0
/* Port E (PORTECFG) */
#define bmGPIFA8     bmBIT7
#define bmT2EX       bmBIT6
#define bmINT6       bmBIT5
#define bmRXD1OUT    bmBIT4
#define bmRXD0OUT    bmBIT3
#define bmT2OUT      bmBIT2
#define bmT1OUT      bmBIT1
#define bmT0OUT      bmBIT0

/* I2C Control & Status Register (I2CS) */
#define bmSTART      bmBIT7
#define bmSTOP       bmBIT6
#define bmLASTRD     bmBIT5
#define bmID         (bmBIT4 | bmBIT3)
#define bmBERR       bmBIT2
#define bmACK        bmBIT1
#define bmDONE       bmBIT0
/* I2C Control Register (I2CTL) */
#define bmSTOPIE     bmBIT1
#define bm400KHZ     bmBIT0
/* Interrupt 2 (USB) Autovector Register (INT2IVEC) */
#define bmIV4        bmBIT6
#define bmIV3        bmBIT5
#define bmIV2        bmBIT4
#define bmIV1        bmBIT3
#define bmIV0        bmBIT2
/* USB Interrupt Request & Enable Registers (USBIE/USBIRQ) */
#define bmEP0ACK     bmBIT6
#define bmHSGRANT    bmBIT5
#define bmURES       bmBIT4
#define bmSUSP       bmBIT3
#define bmSUTOK      bmBIT2
#define bmSOF        bmBIT1
#define bmSUDAV      bmBIT0
/* Breakpoint register (BREAKPT) */
#define bmBREAK      bmBIT3
#define bmBPPULSE    bmBIT2
#define bmBPEN       bmBIT1
/* Interrupt 2 & 4 Setup (INTSETUP) */
#define bmAV2EN      bmBIT3
#define bmINT4IN     bmBIT1
#define bmAV4EN      bmBIT0
/* USB Control & Status Register (USBCS) */
#define bmHSM        bmBIT7
#define bmDISCON     bmBIT3
#define bmNOSYNSOF   bmBIT2
#define bmRENUM      bmBIT1
#define bmSIGRESUME  bmBIT0
/* Wakeup Control and Status Register (WAKEUPCS) */
#define bmWU2        bmBIT7
#define bmWU         bmBIT6
#define bmWU2POL     bmBIT5
#define bmWUPOL      bmBIT4
#define bmDPEN       bmBIT2
#define bmWU2EN      bmBIT1
```

```
#define bmWUEN       bmBIT0
/* End Point 0 Control & Status Register (EP0CS) */
#define bmHSNAK      bmBIT7
/* End Point 0-1 Control & Status Registers (EP0CS/EP1OUTCS/EP1INCS) */
#define bmEPBUSY     bmBIT1
#define bmEPSTALL    bmBIT0
/* End Point 2-8 Control & Status Registers (EP2CS/EP4CS/EP6CS/EP8CS) */
#define bmNPAK       (bmBIT6 | bmBIT5 | bmBIT4)
#define bmEPFULL     bmBIT3
#define bmEPEMPTY    bmBIT2
/* Endpoint Status (EP2468STAT) SFR bits */
#define bmEP8FULL    bmBIT7
#define bmEP8EMPTY   bmBIT6
#define bmEP6FULL    bmBIT5
#define bmEP6EMPTY   bmBIT4
#define bmEP4FULL    bmBIT3
#define bmEP4EMPTY   bmBIT2
#define bmEP2FULL    bmBIT1
#define bmEP2EMPTY   bmBIT0
/* SETUP Data Pointer Auto Mode (SUDPTRCTL) */
#define bmSDPAUTO    bmBIT0
/* Endpoint Data Toggle Control (TOGCTL) */
#define bmQUERYTOGGLE  bmBIT7
#define bmSETTOGGLE    bmBIT6
#define bmRESETTOGGLE  bmBIT5
#define bmTOGCTLEPMASK bmBIT3 | bmBIT2 | bmBIT1 | bmBIT0
/* IBN (In Bulk Nak) enable and request bits (IBNIE/IBNIRQ) */
#define bmEP8IBN     bmBIT5
#define bmEP6IBN     bmBIT4
#define bmEP4IBN     bmBIT3
#define bmEP2IBN     bmBIT2
#define bmEP1IBN     bmBIT1
#define bmEP0IBN     bmBIT0

/* PING-NAK enable and request bits (NAKIE/NAKIRQ) */
#define bmEP8PING     bmBIT7
#define bmEP6PING     bmBIT6
#define bmEP4PING     bmBIT5
#define bmEP2PING     bmBIT4
#define bmEP1PING     bmBIT3
#define bmEP0PING     bmBIT2
#define bmIBN         bmBIT0

/* Interface Configuration bits (IFCONFIG) */
#define bmIFCLKSRC    bmBIT7          // set == INTERNAL
#define bm3048MHZ     bmBIT6          // set == 48 MHz
#define bmIFCLKOE     bmBIT5
#define bmIFCLKPOL    bmBIT4
#define bmASYNC       bmBIT3
#define bmGSTATE      bmBIT2
#define bmIFCFG1      bmBIT1
#define bmIFCFG0      bmBIT0
#define bmIFCFGMASK   (bmIFCFG0 | bmIFCFG1)
#define bmIFGPIF      bmIFCFG1

/* EP 2468 FIFO Configuration bits (EP2FIFOCFG,EP4FIFOCFG,EP6FIFOCFG,EP8FIFOCFG) */
#define bmINFM       bmBIT6
#define bmOEP        bmBIT5
#define bmAUTOOUT    bmBIT4
#define bmAUTOIN     bmBIT3
#define bmZEROLENIN  bmBIT2
```

```
//   must be zero    bmBIT1
#define bmWORDWIDE   bmBIT0

/*
 * Chip Revision Control Bits (REVCTL) - used to ebable/disable revision specific features
 */
#define bmNOAUTOARM    bmBIT1 // these don't match the docs
#define bmSKIPCOMMIT   bmBIT0 // these don't match the docs

#define       bmDYN_OUT      bmBIT1        // these do...
#define       bmENH_PKT      bmBIT0


/* Fifo Reset bits (FIFORESET) */
#define bmNAKALL       bmBIT7

/* Endpoint Configuration (EPxCFG) */
#define       bmVALID               bmBIT7
#define       bmIN                bmBIT6
#define       bmTYPE1               bmBIT5
#define       bmTYPE0               bmBIT4
#define         bmISOCHRONOUS               bmTYPE0
#define         bmBULK               bmTYPE1
#define         bmINTERRUPT              (bmTYPE1 | bmTYPE0)
#define       bm1KBUF               bmBIT3
#define       bmBUF1              bmBIT1
#define       bmBUF0              bmBIT0
#define         bmQUADBUF                0
#define         bmINVALIDBUF               bmBUF0
#define         bmDOUBLEBUF              bmBUF1
#define         bmTRIPLEBUF              (bmBUF1 | bmBUF0)

/* OUTPKTEND */
#define       bmSKIP              bmBIT7        // low 4 bits specify which end point

/* GPIFTRIG defs */
#define       bmGPIF_IDLE             bmBIT7               // status bit

#define       bmGPIF_EP2_START                 0
#define       bmGPIF_EP4_START           1
#define       bmGPIF_EP6_START           2
#define       bmGPIF_EP8_START           3
#define       bmGPIF_READ             bmBIT2
#define       bmGPIF_WRITE                   0

/* EXIF bits */
#define       bmEXIF_USBINT              bmBIT4
#define       bmEXIF_I2CINT              bmBIT5
#define       bmEXIF_IE4              bmBIT6
#define       bmEXIF_IE5              bmBIT7


#endif   /* FX2REGS_H */
```

/lib/i2c.c

```
/* -*- c++ -*- */
/*
 * Copyright 2003 Free Software Foundation, Inc.
 *
 * This file is part of GNU Radio
```

```
 *
 * GNU Radio is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3, or (at your option)
 * any later version.
 *
 * GNU Radio is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with GNU Radio; see the file COPYING.  If not, write to
 * the Free Software Foundation, Inc., 51 Franklin Street,
 * Boston, MA 02110-1301, USA.
 */

#include "i2c.h"
#include "fx2regs.h"
#include <string.h>

// issue a stop bus cycle and wait for completion

// returns non-zero if successful, else 0
unsigned char
i2c_read (unsigned char addr, unsigned char *buf, unsigned char len)
{
  volatile unsigned char     junk;

  if (len == 0)                    // reading zero bytes always works
    return 1;

  while (I2CS & bmSTOP)          // wait for stop to clear
    ;

  I2CS = bmSTART;
  I2DAT = (addr << 1) | 1;        // write address and direction (1's the read bit)

  while ((I2CS & bmDONE) == 0)
    ;

  if ((I2CS & bmBERR) || (I2CS & bmACK) == 0)       // no device answered...
    goto fail;

  if (len == 1)
    I2CS |= bmLASTRD;

  junk = I2DAT;                       // trigger the first read cycle

  while (--len != 0){
    while ((I2CS & bmDONE) == 0)
      ;

    if (I2CS & bmBERR)
      goto fail;

    if (len == 1)
      I2CS |= bmLASTRD;

    *buf++ = I2DAT;          // get data, trigger another read
  }
```

```
  // wait for final byte

  while ((I2CS & bmDONE) == 0)
    ;

  if (I2CS & bmBERR)
    goto fail;

  I2CS |= bmSTOP;
  *buf = I2DAT;

  return 1;

 fail:
  I2CS |= bmSTOP;
  return 0;
}

// returns non-zero if successful, else 0
unsigned char
i2c_write (unsigned char addr, unsigned char *buf, unsigned char len)
{
  while (I2CS & bmSTOP)         // wait for stop to clear
    ;

  I2CS = bmSTART;
  I2DAT = (addr << 1) | 0;        // write address and direction (0's the write bit)

  while ((I2CS & bmDONE) == 0)
    ;

//  if ((I2CS & bmBERR) || (I2CS & bmACK) == 0)       // no device answered...
  //  goto fail;

  while (len > 0){
    I2DAT = *buf++;
    len--;

    while ((I2CS & bmDONE) == 0)
      ;

    if ((I2CS & bmBERR) || (I2CS & bmACK) == 0)     // no device answered...
      goto fail;
  }

  I2CS |= bmSTOP;
  return 1;

 fail:
  I2CS |= bmSTOP;
  return 0;
}
```

To compile this project use `build.bat` as showed above. Compared to the GNU radio I changed the I2C functions from datatype `(const) xdata unsigned char *buf` to `unsigned char *buf`.

Download the firmware

Windows XP

Under Windows you can use the download tool from Cypress. After the installation of the Cypress DevKit (see Installing the tools above) it can be found at Start Programs Cypress USB CyConsole EZ-USB. Connect the GECKO3 or the development bard to an USB port. The EZ-USB Interface tool should recognize this and "USB Device" appears at the left upper corner. If you only want to test your firmware use the download button and select the `*.hex file`. This program is gone after a restart. In case you want to burn your firmware into the EEPROM (do this only after testing!) use the S EEPROM or L EEPROM button and select the `*.iic` file.

Linux

`fxload -I fw.hex` (Kernel 2.6), with 2.4 you have to use this driver

Search the bus and device number of the connected Cypress chip with:

```
sudo lsusb
```

After that you can download the firmware into the RAM of the Cypress chip by this command:

```
sudo fxload -t fx2 -I /<path to firmware file>/firmware.hex -D /proc/bus/usb/<bus number>/<device number>
```

If your programm is bigger than 8 kByte then you have to use an A3 loader with fxload, use this command instead of the above one:

```
sudo fxload -t fx2 -s /usr/share/usb/a3load.hex -I /<path to firmware file>/firmware.hex -D /proc/bus/usb/<bus number>/<device number>
```

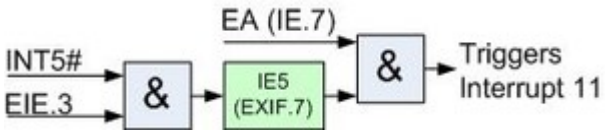
During development you have to adjust the access rights manually.
The device number changes after firmware download, use again `lsusb` to search the new device number!

To give read and write access to everybody:

```
sudo chmod 666 /proc/bus/usb/<bus number>/<device number>
```

The normal way to manage access rights for devices is through a udev rule. Writing udev rules

Demo Firmware: Read an external IRQ



- EA has to be 1 to enable IRQs in general
- A negative edge at INT5# sets IE5 high, this triggers the ISR
- IE5 has to be cleared in the ISR
- Set IE5 high in the Code triggers a soft IRQ and triggers the ISR
- EIE.3 = Enable external IRQ5
- The priority can be set with PX5 (EIP.3): 0 = Low; 1 = High

```
/* Copyright (C) 2008 by Christoph Zimmermann and Lukas Kohler
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
```

```
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */


#define ALLOCATE_EXTERN
#include <fx2regs.h>
#include <i2c.h>

#define BLOCKSIZE 64

#define PORTOUTA    IOA
#define PORTOUTB    IOB
#define PORTOUTCFGA OEA
#define PORTOUTCFGB OEB
#define PORTIN      IOD
#define PORTINCFG   OED
#define PORTIND     PD0

#define ADD_SW      0x20
#define ADD_7SEG    0x21

void isrExt5(void) interrupt 11;

/*****************************************************************************/
/*  \fn         void isrExt5(void) interrupt 11                              */
/*****************************************************************************/
/*! \brief      ISR for External IRQ5
 ******************************************************************************
 *  \param[in]  -
 *  \param[out] -
 *  \return     -
 ******************************************************************************
 *  \author     Lukas Kohler
 *  \date       29.07.2007   LK created
 *  \test       OK
 *****************************************************************************/
void isrExt5(void) interrupt 11
{
  int i=0;
  EA = 0;               // disable IRQs
  EXIF &= 0x7F;         // clear irq

  PORTOUTB = 0xFF;
  for(i=0;i<0xFFFF;i++){;}
  PORTOUTB = 0x00;
  for(i=0;i<0xFFFF;i++){;}

  EA = 1;               // enable IRQs
}

/*****************************************************************************/
/*  \fn         void init_fx2(void)                                          */
/*****************************************************************************/
/*! \brief      Initializes the FX2 contoller
 ******************************************************************************
 *  \param[in]  -
 *  \param[out] -
 *  \return     -
 ******************************************************************************
 *  \author     Lukas Kohler
 *  \date       17.07.2007   LK created
```

```
 *  \test       OK
 *****************************************************************************/
void init_fx2(void)
{
  CPUCS = 0x08;       // 24MHz CPU-Clock
  EP1OUTBC = 0x01;    // Writing to EP1OUTBC rearms for an out Transfer
  PORTOUTCFGA = 0xff; // Port A (PA) is output
  PORTOUTCFGB = 0xff; // Port B (PB) is output
  PORTOUTA = 0x00;    // reset PA
  PORTOUTB = 0x00;    // reset PA
  PORTINCFG = 0xff;   // Eingabeport auf Ausgabe
  PORTIN = 0x00;      // Einmal auf Null stellen
  PORTINCFG = 0x00;   // Jetzt Eingabeport auf Eingabe einstellen
  // IRQ's
  EIEX5 = 1;          // Enable external interrupt 5.
  EA = 1;             // Enable global IRQs
}

/*****************************************************************************/
/*  \fn         void main(void)                                              */
/*****************************************************************************/
/*! \brief      Main routine in endless loop. IRQ example on Ext IRQ5.
 *              On the Cypress development board shows this I2C data a
 *              counter on the 7segment display. A pressed SW1 is reset
 *              of the counter. A falling edge on IRQ5 will trigger an
 *              IRQ and turn on PB.
 *****************************************************************************
 *  \param[in]  -
 *  \param[out] -
 *  \return     -
 *****************************************************************************
 *  \author     Lukas Kohler
 *  \date       18.07.2007   LK created
 *  \date       29.07.2007   LK added ISR and removed running light
 *  \test       OK
 *****************************************************************************/

void main(void)
{
  bit p=0;
  unsigned char cmd[1];
  unsigned char Digit[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x98, 0x88, 0x83, 0xc6,
0xa1, 0x86, 0x8e };
  // 7 segment data        0     1     2     3     4     5     6     7     8     9     A     B     C     D
E     F

  long x;
  unsigned char i;

  init_fx2(); // initialize FX2
  while(1)    // do endless
  {
    for(i=0; i<16; ) // Counter 0..15
    {
      cmd[0] = Digit[i];
      i2c_write(ADD_7SEG, cmd, 1);   // write data to the 7segment controller

      i++;
       #pragma noinduction
      for(x=0; x<2000; x++){         // have a break - have a kitkat
        i2c_read (ADD_SW, cmd, 1);   // delay with reading switche SW1
```

```
        if(!(cmd[0]& 0x01)){
          i = 0;   // reset counter if SW1 is pressed
        }
      }
    }
  }
}
```

USB vendor and product IDs

Every device on the USB bus has an unique device identifier (ID). The vendor ID of the GECKO3main is 0x0547 and the product ID 0x8888. This vendor ID does according usb not jet exist. Windows stores the information about its installed USB devices in the registry under `[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\Vid_0547&Pid_8888]`

	vendor ID	product ID
EEPROM small	0x04B4	0x8613
EEPROM large	0x04B4	0x0001

A tool to view the USB traffic is USB sniffer. This tool allows as well to uninstall a corrupted driver from Windows (button Replug). A short introduction can be found under Different Tool hints.

Optimizing Code

Working on 8 bit microcontrollers and programming in C for a system with only 16 kbyte of memory needs some knowledge how to write effiecient code for this environment. A good overview over many important coding tricks is this article from Jones, Nigel. "Efficient C Code for Eight-Bit MCUs" Embedded Systems Programming, November 1998.