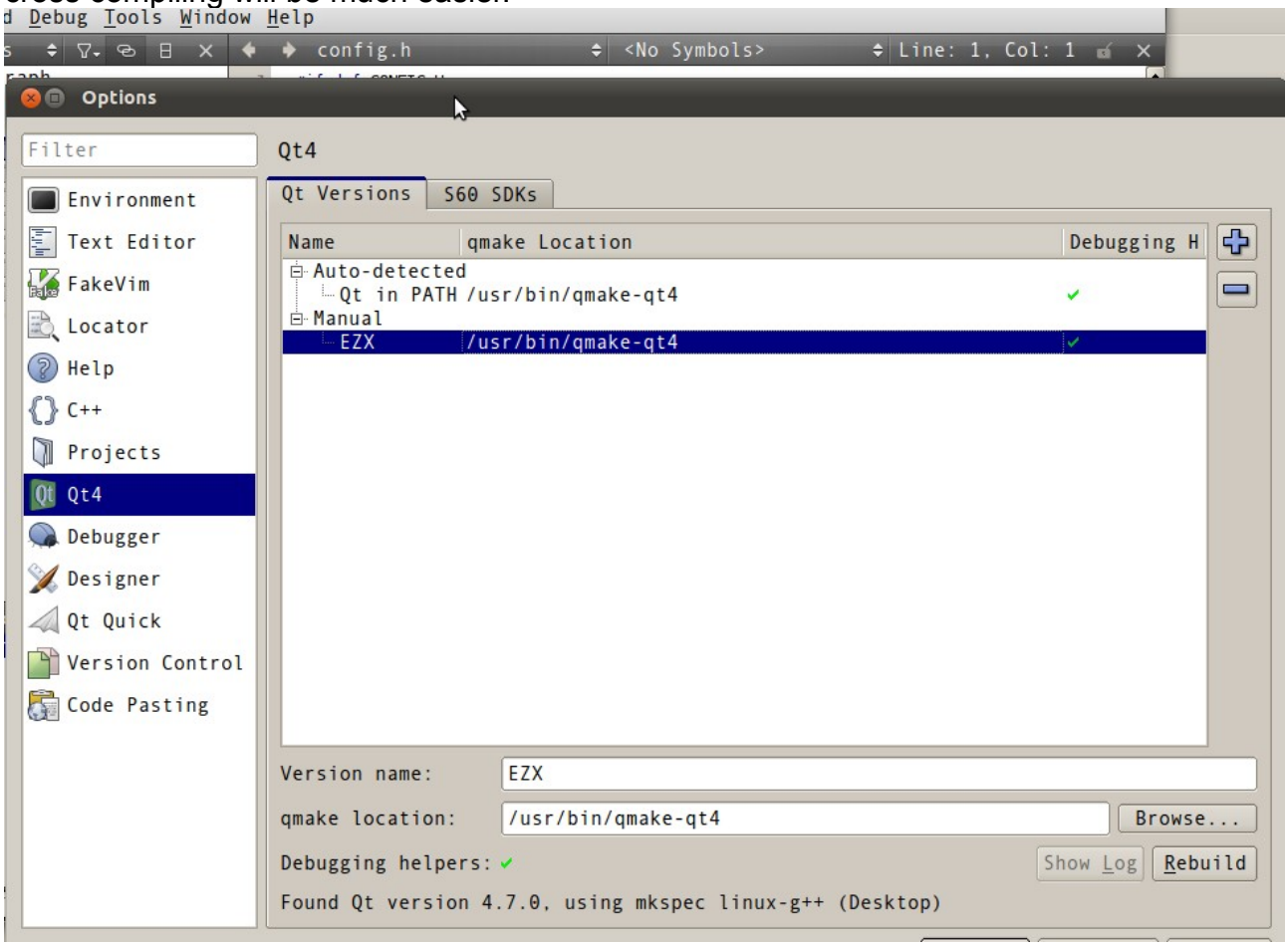
Cross Compile With Qt Creator

Wang Bin 2011-02-17
wbsecg1@gmail.com

Qt Creator is a great IDE for building Qt4.x applications. I had an crazy idea. Can we use it for cross compiling or compiling other version of qt applications? After some tests, I found that the answer is yes!

Here I just tell you how to add Motorola's EZX development environment (montavista+qte2.3.8) to your Qt Creator.

First, start Qt Creator. Click Tools menu on the top bar. Select Options item. A dialog will show. Select Qt4 item and add a version in Qt Versions-Manual. Versions name is EZX. Qmake location is /usr/bin/qmake-qt4. You can't use qmake-qt3, because qt creator can't read the versions and can't run qmake. Qmake-qt4 is ok. But the problem is that qt creator will use mkspec linux-g++ (Desktop). What we want is linux-g++-montavista(suppose that we have this mkspec)! I don't know how to change the default mkspec here. If we can, cross compiling will be much easier.



Now I will tell you how to write a mkspec. You must add a folder in mkspec directory, which is /usr/share/qt4/mkspecs in my laptop. The folder name is linux-g++-montavista. Then write qmake.conf and qplatformdefs.h. You can read the existing files for reference.

Here is my qmake.conf:

```
#
#
# qmake configuration for linux-g++-montavista
```

```
#

MAKEFILE_GENERATOR      = UNIX
TARGET_PLATFORM         = unix #Qt4
TEMPLATE        = app
CONFIG                  += qt warn_off release
#incremental link_prl
QMAKE_INCREMENTAL_STYLE = sublib

#
# qmake configuration for common gcc
#

QMAKE_CC                = $(CCACHE) $(DISTCC) iwmmxt_le-gcc
QMAKE_CFLAGS                = -pipe
QMAKE_CFLAGS_DEPS       = -M
QMAKE_CFLAGS_WARN_ON        = -Wall -W
QMAKE_CFLAGS_WARN_OFF       = -w
QMAKE_CFLAGS_RELEASE = -O2 -mcpu=iwmmxt -mtune=iwmmxt
QMAKE_CFLAGS_DEBUG    = -g
QMAKE_CFLAGS_SHLIB      = -fPIC
QMAKE_CFLAGS_STATIC_LIB      += -fPIC #Qt4
QMAKE_CFLAGS_YACC       = -Wno-unused -Wno-parentheses
QMAKE_CFLAGS_THREAD   = -D_REENTRANT  #Qt4 +=

#Qt4
#QMAKE_CFLAGS_HIDESYMS   += -fvisibility=hidden
#QMAKE_CFLAGS_PRECOMPILE += -x c-header -c ${QMAKE_PCH_INPUT}
-o ${QMAKE_PCH_OUTPUT}
#QMAKE_CFLAGS_USE_PRECOMPILE += -include $
{QMAKE_PCH_OUTPUT_BASE}

QMAKE_CXX               = $(CCACHE) $(DISTCC) iwmmxt_le-g++
QMAKE_CXXFLAGS              = $$QMAKE_CFLAGS  -DQWS -fno-exceptions
-fno-rtti
QMAKE_CXXFLAGS_DEPS    = $$QMAKE_CFLAGS_DEPS
QMAKE_CXXFLAGS_WARN_ON    = $$QMAKE_CFLAGS_WARN_ON
QMAKE_CXXFLAGS_WARN_OFF   = $$QMAKE_CFLAGS_WARN_OFF
QMAKE_CXXFLAGS_RELEASE     = $$QMAKE_CFLAGS_RELEASE
QMAKE_CXXFLAGS_DEBUG= $$QMAKE_CFLAGS_DEBUG
QMAKE_CXXFLAGS_SHLIB  = $$QMAKE_CFLAGS_SHLIB
QMAKE_CXXFLAGS_YACC    = $$QMAKE_CFLAGS_YACC
QMAKE_CXXFLAGS_THREAD       = $$QMAKE_CFLAGS_THREAD  #Qt4 +=

QMAKE_INCDIR          = $(MONTAVISTA)/target/usr/include $
(MONTAVISTA)/target/usr/local/include
QMAKE_LIBDIR          = $(MONTAVISTA)/target/usr/lib $
(MONTAVISTA)/target/usr/lib $(MONTAVISTA)/target/usr/local/lib
QMAKE_INCDIR_X11   = /usr/X11R6/include
QMAKE_LIBDIR_X11    = /usr/X11R6/lib
QMAKE_INCDIR_QT             = $(QTDIR)/include $(EZXDIR)/include $
(QT_EXTDIR)/include
```

```
QMAKE_LIBDIR_QT           = $(QTDIR)/lib $(EZXDIR)/lib $(QT_EXTDIR)/lib
QMAKE_INCDIR_OPENGL    = /usr/X11R6/include
QMAKE_LIBDIR_OPENGL    = /usr/X11R6/lib

QMAKE_LINK             = iwmmxt_le-g++
QMAKE_LINK_SHLIB  = iwmmxt_le-g++
QMAKE_LINK_C           = iwmmxt_le-gcc #Qt4
QMAKE_LINK_C_SHLIB       = iwmmxt_le-gcc #Qt4
QMAKE_LFLAGS           =  #Qt4 +=
QMAKE_LFLAGS_RELEASE =  #Qt4 +=
QMAKE_LFLAGS_DEBUG     =  #Qt4 +=
QMAKE_LFLAGS_APP  += #Qt4
QMAKE_LFLAGS_SHLIB       = -shared  #Qt4 +=
QMAKE_LFLAGS_PLUGIN    = $$QMAKE_LFLAGS_SHLIB  #Qt4 +=
QMAKE_LFLAGS_SONAME  = -Wl,-soname,  #Qt4 +=
QMAKE_LFLAGS_THREAD   =  #Qt4 +=
QMAKE_RPATH            = -Wl,-rpath,
QMAKE_LFLAGS_RPATH = -Wl,-rpath,
#Qt4: QMAKE_LFLAGS_RPATH

QMAKE_PCH_OUTPUT_EXT    = .gch

# -Bsymbolic-functions (ld) support
QMAKE_LFLAGS_BSYMBOLIC_FUNC = -Wl,-Bsymbolic-functions
QMAKE_LFLAGS_DYNAMIC_LIST = -Wl,--dynamic-list,

#
# qmake configuration for common linux
#

QMAKE_LIBS             =
QMAKE_LIBS_DYNLOAD      = -ldl
QMAKE_LIBS_X11             = -lXext -lX11 -lm
QMAKE_LIBS_X11SM  = -lSM -lICE
QMAKE_LIBS_NIS             = -lnsl
QMAKE_LIBS_QT              =  -lqte-mt -lezxappsdk -lipp-jp -lezxopenwindow
-lipp-miscGen -lezxappbase -lezxjpeg -lezxpm
QMAKE_LIBS_QT_THREAD  = -lpthread -lqte-mt -lezxappsdk -lipp-jp
-lezxopenwindow -lipp-miscGen -lezxappbase -lezxjpeg -lezxpm
QMAKE_LIBS_OPENGL       = -lGLU -lGL -lXmu
QMAKE_LIBS_OPENGL_QT  = -lGL -lXmu
QMAKE_LIBS_THREAD       = -lpthread -lqte-mt -lezxappsdk -lipp-jp
-lezxopenwindow -lipp-miscGen -lezxappbase -lezxjpeg -lezxpm

QMAKE_MOC              = $(QTDIR)/bin/moc
QMAKE_UIC              = $(QTDIR)/bin/uic

QMAKE_AR               = iwmmxt_le-ar cqs
QMAKE_RANLIB           =

QMAKE_TAR              = tar -cf
QMAKE_GZIP             = gzip -9f
```

```
QMAKE_COPY              = cp -f
QMAKE_COPY_FILE         = $(COPY)
QMAKE_COPY_DIR          = $(COPY) -r
QMAKE_MOVE              = mv -f
QMAKE_DEL_FILE          = rm -f
QMAKE_DEL_DIR           = rmdir
QMAKE_STRIP             = iwmmxt_le-strip
QMAKE_STRIPFLAGS_LIB    += --strip-unneeded
QMAKE_CHK_DIR_EXISTS    = test -d
QMAKE_MKDIR             = mkdir -p

#
# qmake configuration for common unix
#

QMAKE_LEX               = flex
QMAKE_LEXFLAGS          +=
QMAKE_YACC              = yacc
QMAKE_YACCFLAGS         += -d
QMAKE_YACCFLAGS_MANGLE  += -p $base -b $base
QMAKE_YACC_HEADER       = $base.tab.h
QMAKE_YACC_SOURCE       = $base.tab.c
QMAKE_PREFIX_SHLIB      = lib
QMAKE_PREFIX_STATICLIB  = lib
QMAKE_EXTENSION_STATICLIB = a
```

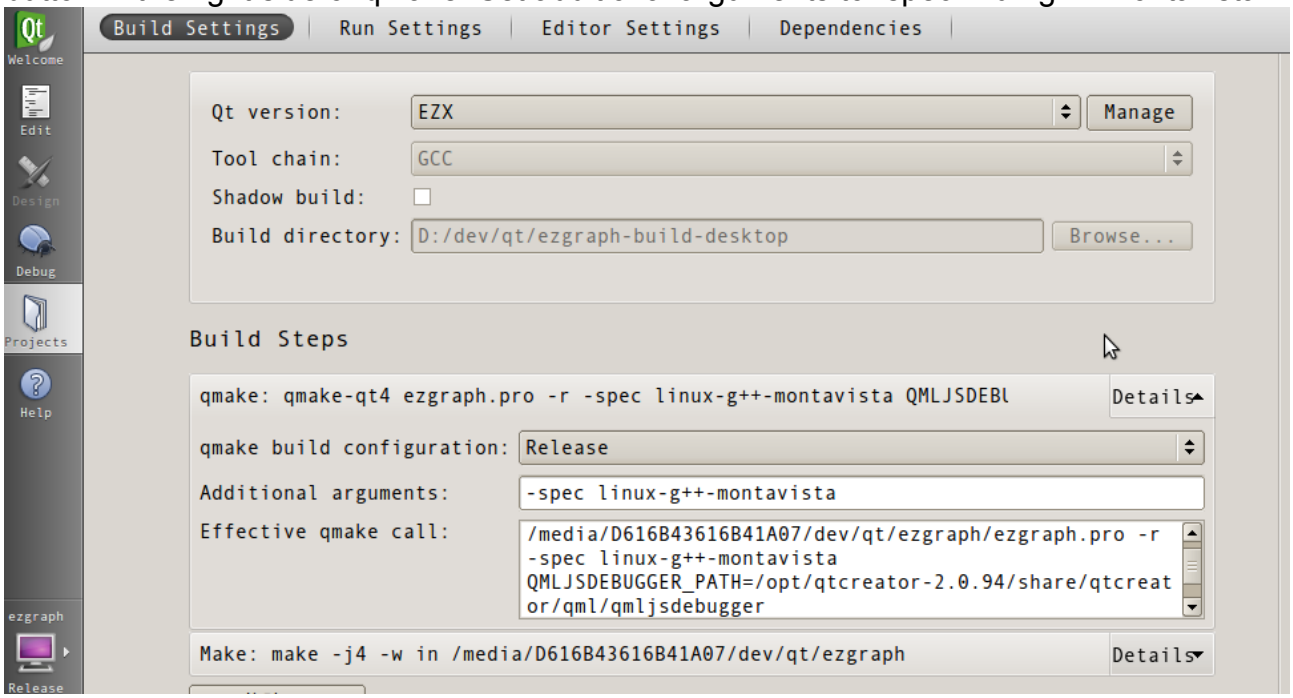
And qplatformdefs.h:

```
#ifndef QPLATFORMDEFS_H
#define QPLATFORMDEFS_H

// Get Qt defines/settings

#include "qglobal.h"

// Set any POSIX/XOPEN defines at the top of this file to turn on specific APIs

// DNS system header files are a mess!
// <resolv.h> includes <arpa/nameser.h>. <arpa/nameser.h> is using
// 'u_char' and includes <sys/types.h>.  Now the problem is that
// <sys/types.h> defines 'u_char' only if __USE_BSD is defined.
// __USE_BSD is defined in <features.h> if _BSD_SOURCE is defined.
#ifndef _BSD_SOURCE
#  define _BSD_SOURCE
#endif

// 1) need to reset default environment if _BSD_SOURCE is defined
// 2) need to specify POSIX thread interfaces explicitly in glibc 2.0
// 3) it seems older glibc need this to include the X/Open stuff
#ifndef _GNU_SOURCE
#  define _GNU_SOURCE
#endif
```

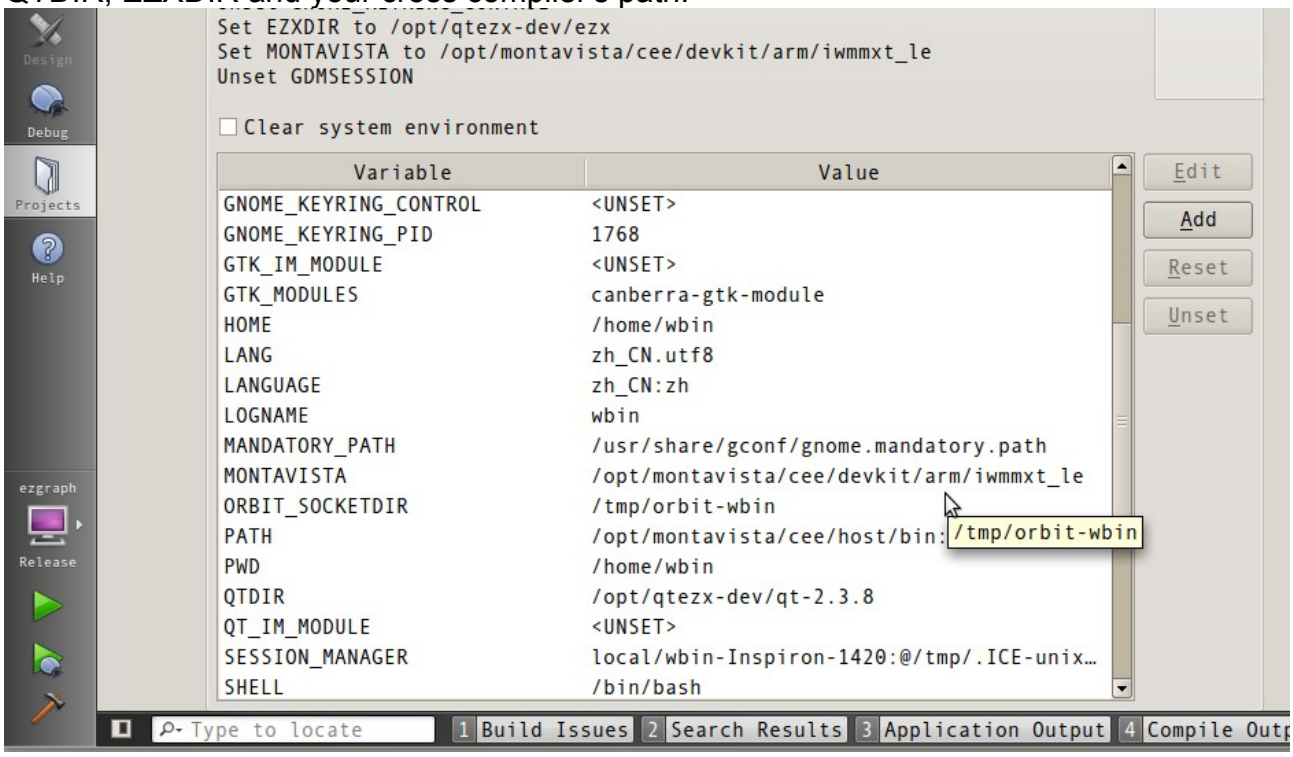
```
#include <unistd.h>

// We are hot - unistd.h should have turned on the specific APIs we requested

#ifdef QT_THREAD_SUPPORT
#include <pthread.h>
#endif

#include <dirent.h>
#include <fcntl.h>
#include <grp.h>
#include <pwd.h>
#include <signal.h>
#include <dlfcn.h>

#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/ipc.h>
#include <sys/time.h>
#include <sys/shm.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <sys/wait.h>

// DNS header files are not fully covered by X/Open specifications.
// In particular nothing is said about res_* :/
// Header files <netinet/in.h> and <arpa/nameser.h> are not included
// by <resolv.h> on older versions of the GNU C library. Note that
// <arpa/nameser.h> must be included before <resolv.h>.
#include <netinet/in.h>
#include <arpa/nameser.h>
#include <resolv.h>

#if !defined(QT_NO_COMPAT)
#define QT_STATBUF              struct stat
#define QT_STATBUF4TSTAT        struct stat
#define QT_STAT                 ::stat
#define QT_FSTAT        ::fstat
#define QT_STAT_REG             S_IFREG
#define QT_STAT_DIR             S_IFDIR
#define QT_STAT_MASK            S_IFMT
#define QT_STAT_LNK             S_IFLNK
#define QT_FILENO               fileno
#define QT_OPEN                 ::open
#define QT_CLOSE                ::close
#define QT_LSEEK        ::lseek
#define QT_READ                 ::read
#define QT_WRITE        ::write
```

```
#define QT_ACCESS               ::access
#define QT_GETCWD               ::getcwd
#define QT_CHDIR        ::chdir
#define QT_MKDIR        ::mkdir
#define QT_RMDIR        ::rmdir
#define QT_OPEN_RDONLY              O_RDONLY
#define QT_OPEN_WRONLY              O_WRONLY
#define QT_OPEN_RDWR                O_RDWR
#define QT_OPEN_CREAT               O_CREAT
#define QT_OPEN_TRUNC               O_TRUNC
#define QT_OPEN_APPEND              O_APPEND
#endif

#define QT_SIGNAL_RETTYPE  void
#define QT_SIGNAL_ARGS              int
#define QT_SIGNAL_IGNORE    SIG_IGN

#if defined(__GLIBC__) && (__GLIBC__ >= 2)
#define QT_SOCKLEN_T        socklen_t
#else
#define QT_SOCKLEN_T        int
#endif

#if defined(_XOPEN_SOURCE) && (_XOPEN_SOURCE >= 500)
#define QT_SNPRINTF             ::snprintf
#define QT_VSNPRINTF            ::vsnprintf
#endif


#endif // QPLATFORMDEFS_H
```

Now in qt creator's Project item, You can choose Qt versions “EZX”! The press Details button in the right side of qmake. Set additional arguments to -spec linux-g++-montavista.



The last step is change some environment variables in Build Environment. For example QTDIR, EZXDIR and your cross compiler's path.



If your configuration is right, then you can cross compile using qt creator.

This is not a perfect way. You must setup the mkspec and environment for every project. The argument QMLJSDEBUGGER_PATH is unnecessary but can't be removed. The Makefile generated by qmake-qt4 will including some unnecessary variables such as DIST=xxx.prf. But there's no problem to cross compile in this way!

Try if yourself!

My ezx sdk project: http://code.google.com/p/moto-e6-sdk/