Compiler Integration with Yocto Project* 1.3 Application Development Toolkit

This article describes how to use the Intel® C++ Compiler 13.x from the Intel System Studio for Linux. Please refer to http://software.intel.com/en-us/intel-system-studio for more details regarding Intel System Studio for Linux product.

Yocto Project* Application Development Toolkit (ADT) Installer

Please refer to the Yocto Project* Application Development Toolkit User's Guide at http://www.yoctoproject.org/docs/current/adt-manual/adt-manual.html for details on the Application Development Toolkit Usage.

To be able to use the Application Development Toolkit with the Intel® C++ Compiler it has to be installed using the Application Development Toolkit tarball and installer script available at http://downloads.yoctoproject.org/releases/yocto/yocto-1.3/adt_installer/. More detailed information on this can be found at

http://www.yoctoproject.org/docs/current/adt-manual/adt-manual.html#using-the-adt-installer.

After unpacking the installer tarball you will find a file adt_installer.conf. This file needs to be modified to support Intel® architecture before the Application Development Toolkit can be used with the Intel® C++ Compiler.

Modifying the Application Development Toolkit Installer Configuration

In the file adt_installer.conf the following modifications are necessary:

1. Remove arm in line 31

   YOCTOADT_TARGETS="x86"

2. Comment out lines 43, 45, and 47

   #YOCTOADT_ROOTFS_arm="minimal sato-sdk"

   #YOCTOADT_TARGET_SYSROOT_IMAGE_arm="sato-sdk"

   #YOCTOADT_TARGET_SYSROOT_LOC_arm="$HOME/test-yocto/arm"

3. Uncomment lines 50, 51, and 52

   YOCTOADT_ROOTFS_x86="sato-sdk"

   YOCTOADT_TARGET_SYSROOT_IMAGE_x86="sato-sdk"

   YOCTOADT_TARGET_SYSROOT_LOC_x86="$HOME/test-yocto/x86"

The important changes are that the three entries

   YOCTOADT_ROOTFS_x86

   YOCTOADT_TARGET_SYSROOT_IMAGE_x86

   YOCTOADT_TARGET_SYSROOT_LOC_x86

need to be defined.

These entries are used to configure the Application Development Toolkit environment and need to be present for the Intel C++ Compiler for Intel® Atom™ processor to integrate into this

environment.

Run the Application Development Toolkit Installer

Now the Application Development Toolkit can be installed. Simply execute

```
$ adt_installer &
```

from the unpacked installer tarfile and follow the instructions.

Integrate Intel C++ Compiler with the Application Development Toolkit

Copy the existing file /opt/poky/1.3/environment-setup-i586-poky-linux to /opt/poky/1.3/environment-setup-i586-poky-linux-icc, and make the following changes in the new file. The changes are marked as red:

```
source /opt/intel/system_studio_2013.1.024/bin/iccvars.sh ia32
export YL_TOOLCHAIN=/opt/poky/1.3/sysroots/x86_64-pokysdk-linux/usr/bin
export YL_SYSROOT= =<CURRENT_USER>/test-yocto/x86
export
PATH=/opt/poky/1.3/sysroots/x86_64-pokysdk-linux/usr/bin:/opt/poky/1.3/sysroots/x86_64-pok
ysdk-linux/usr/bin/i586-poky-linux:$PATH
export PKG_CONFIG_SYSROOT_DIR=<CURRENT_USER>/test-yocto/x86
export PKG_CONFIG_PATH=<CURRENT_USER>/test-yocto/x86/usr/lib/pkgconfig
export CONFIG_SITE=/opt/poky/1.3/site-config-i586-poky-linux
export CC="icc -platform=yl13 -m32 -march=i586 "
export CXX="icpc -platform=yl13 -m32 -march=i586 "
export CPP="i586-poky-linux-gcc -E -m32 -march=i586
--sysroot=<CURRENT_USER>/test-yocto/x86"
export AS="i586-poky-linux-as "
export LD="xild -qplatform=yl13 "
export GDB=i586-poky-linux-gdb
export STRIP=i586-poky-linux-strip
export RANLIB=i586-poky-linux-ranlib
export OBJCOPY=i586-poky-linux-objcopy
export OBJDUMP=i586-poky-linux-objdump
export AR="xiar -qplatform=yl13"
export NM=i586-poky-linux-nm
export TARGET_PREFIX=i586-poky-linux-
export CONFIGURE_FLAGS="--target=i586-poky-linux --host=i586-poky-linux --build=x86_64-linux
--with-libtool-sysroot=<CURRENT_USER>/test-yocto/x86"
export CFLAGS=" -O2 -pipe -g "
export CXXFLAGS=" -O2 -pipe -g -fpermissive"
export LDFLAGS="-Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed"
export CPPFLAGS=""
export OECORE_NATIVE_SYSROOT="/opt/poky/1.3/sysroots/x86_64-pokysdk-linux"
export OECORE_TARGET_SYSROOT="/home/ywang30/test-yocto/x86"
export OECORE_ACLOCAL_OPTS="-I
/opt/poky/1.3/sysroots/x86_64-pokysdk-linux/usr/share/aclocal"
export OECORE_DISTRO_VERSION="1.3"
```

export OECORE_SDK_VERSION="1.3"
The –sysroot compile flags from the original environment-setup-i586-poky-linux can be removed because their function is covered by the YL_SYSROOT environment variable which will be used by Intel C++ compiler.
Intel C++ compiler is compatible with gcc compiler. In the file /opt/poky/1.3/environment-setup-i586-poky-linux-icc, we add 3 lines to set up the icc working environment. We changed 4 lines including the definition for CC, CXX, LD and AR. We removed the option "-feliminate-unused-debug-types" from CFLAGS and CXXFLAGS since it is not supported by icc in order to eliminate the warning during icc compilation.

Setup of Compiler Build Environment

To setup the environment for the Intel® C++ Compiler on your Linux host, just execute the following command:

> source /opt/poky/1.3/environment-setup-i586-poky-linux-icc

Now you are ready to use the Intel® C++ Compiler for any Yocto Project* targeted application that uses the Poky* Linux* based Application Development Toolkit* (ADT) framework.

If you are using a Cross-Toolchain Tarball as mentioned in http://www.yoctoproject.org/docs/current/adt-manual/adt-manual.html#using-an-existing-toolchain-tarball, you can make the similar changes to file /opt/poky/1.3/environment-setup-i586-poky-linux and make the icc works with poky cross toolchain.

The document also applies for Yocto Project* 1.2 application development toolkit. You need to change the icc option -platform=yl13 to -platform=yl12 for Yocto 1.2 integration.

Use improved sysroot support in Intel C++ compiler 14.0

In the Intel C++ compiler version 14.0, which released in the Intel System Studio 2014 Beta, we can use option –gnu-prefix and --sysroot for the cross compile. For details of the options, please refer to http://software.intel.com/en-us/articles/improved-sysroot-support-in-intel-c-compiler-for-cross-compile

With the Intel C++ compiler 14.0, we can change the file /opt/poky/1.3/environment-setup-i586-poky-linux-icc as following. The changes are marked as red bold. We don't need to export the environment variable in this case. We don't need the option –platform as well. We just need to change the compiler name and add option –gnu-prefix.

source /opt/intel/system_studio_2013.1.024/bin/iccvars.sh ia32
export
PATH=/opt/poky/1.3/sysroots/x86_64-pokysdk-linux/usr/bin:/opt/poky/1.3/sysroots/x86_64-pokysdk-linux/usr/bin/i586-poky-linux:$PATH
export PKG_CONFIG_SYSROOT_DIR=<CURRENT_USER>/test-yocto/x86

export PKG_CONFIG_PATH=<CURRENT_USER>/test-yocto/x86/usr/lib/pkgconfig
export CONFIG_SITE=/opt/poky/1.3/site-config-i586-poky-linux
export CC="**icc -gnu-prefix=i586-poky-linux-** -m32 -march=i586 --sysroot=/opt/poky/1.3/sysroots/i586-poky-linux"
export CXX="**icpc -gnu-prefix=i586-poky-linux-** -m32 -march=i586 --sysroot=/opt/poky/1.3/sysroots/i586-poky-linux"export AS="i586-poky-linux-as "
export LD= "**xild -qgnu-prefix=i586-poky-linux-** --sysroot=/opt/poky/1.3/sysroots/i586-poky-linux"
export GDB=i586-poky-linux-gdb
export STRIP=i586-poky-linux-strip
export RANLIB=i586-poky-linux-ranlib
export OBJCOPY=i586-poky-linux-objcopy
export OBJDUMP=i586-poky-linux-objdump
export AR="**xiar -qgnu-prefix=i586-poky-linux-** "
export NM=i586-poky-linux-nm
export TARGET_PREFIX=i586-poky-linux-
export CONFIGURE_FLAGS="--target=i586-poky-linux --host=i586-poky-linux --build=x86_64-linux --with-libtool-sysroot=<CURRENT_USER>/test-yocto/x86"
export CFLAGS=" -O2 -pipe -g "
export CXXFLAGS=" -O2 -pipe -g -fpermissive"
export LDFLAGS="-Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed"
export CPPFLAGS=""
export OECORE_NATIVE_SYSROOT="/opt/poky/1.3/sysroots/x86_64-pokysdk-linux"
export OECORE_TARGET_SYSROOT="/home/ywang30/test-yocto/x86"
export OECORE_ACLOCAL_OPTS="-I /opt/poky/1.3/sysroots/x86_64-pokysdk-linux/usr/share/aclocal"
export OECORE_DISTRO_VERSION="1.3"
export OECORE_SDK_VERSION="1.3"