EDK II Standard C Library
ReadMe
Alpha 2 Release

This document describes the EDK II specific aspects of installing, building, and using the Standard C Library component of the Intel® UEFI Application Development Kit, (Intel® UADK).

This release of the Intel® UADK has some restrictions, as described below.

1. Only the Microsoft VS2005 and VS2008, Intel C Compiler 10.1 (or later), GCC 4.3 (mingw32), GCC 4.4, and GCC 5.4 C compilers are supported for Ia32 or X64 CPU architectures.
2. The target machine must be running EDK II based firmware with the EDK II HII present and enabled.
3. The Intel® UADK has not been through Intel's Quality Assurance process. This means that specified standards compliance has not been validated, nor has it undergone formal functionality testing.

The Standard C Library provided by this package is a "hosted" implementation conforming to the ISO/IEC 9899-1990 C Language Standard with Addendum 1. This is commonly referred to as the "C 95" specification.

The following instructions assume that you have an existing EDK II or UDK 2010 source tree that has been configured to build with your tool chain.

The UADK is comprised of three packages: AppPkg, StdLib, and StdLibPrivateInternalFiles.

AppPkg — This package contains applications which demonstrate use of the Standard C Library. These applications reside in `AppPkg/Applications`.

- Enquire — This is a program that determines many properties of the C compiler and the target machine that Enquire is run on. The only changes required to port this 1990s era Unix program to EDK II were the addition of 8 pragmas to enquire.c in order to disable some Microsoft VC++ specific warnings.
- Hello — This is a very simple EDK II native application that doesn't use any features of the *Standard C Library*.
- Main — This application is functionally identical to Hello, except that it uses the *Standard C Library* to provide a main() entry point.

StdLib — The StdLib package contains the standard header files as well as implementations of the standard libraries. Currently, the only standard library provided is the *Standard C Library*, the implementation of which is in the `StdLib/LibC` directory.

StdLibPrivateInternalFiles The contents of this package are for the exclusive use of the library implementations in StdLib. Please do not use anything from this package in your application or unexpected behavior may occur. This package may be removed from a future release.

INSTALLATION

Install the UADK packages by extracting, downloading or copying them to the root of your EDK II source tree. The three package directories should be peers to the Conf, MdePkg, Nt32Pkg, etc. directories.

BUILDING

It is not necessary to build the libraries separately from the target application(s). If the application references the libraries, as described in USAGE; below; the required libraries will be built as needed.

To build the applications included in AppPkg, one would execute the following commands within the "Visual Studio Command Prompt" window:

```
> cd C:\Source\Edk2
> .\edksetup.bat
> build –a X64 –p AppPkg\AppPkg.dsc
```

This will produce the application executables: Enquire.efi, Hello.efi, and Main.efi in the `C:\Source\Edk2\Build\AppPkg\DEBUG_VS2008\X64` directory; with the DEBUG_VS2008 component being replaced with the actual tool chain and build type you have selected in Conf\Tools_def.txt. These executables can now be loaded onto the target platform and executed.

If you examine the AppPkg.dsc file, you will notice that the StdLib package is referenced in order to resolve the library classes comprising the *Standard C Library*. This, plus referencing the StdLib package in your application's .inf file is all that is needed to link your application to the standard libraries.

USAGE

This implementation of the Standard C Library is comprised of 12 separate libraries in addition to the standard header files. Each library is associated with use of one of the standard headers; thus, if the header is used in an application, it must be linked with the associated library. The associations are described in the following table.

Library Class	Header File(s)	Notes
LibC	-- Use Always --	This library is always required.
LibUefi	sys/EfiSysCall.h	Provides the UEFI system interface and "System Calls"
LibStdLib	stdlib.h	
LibString	string.h	

Library Class	Header File(s)	Notes
LibSignal	signal.h	
LibStdio	stdio.h	
LibWchar	wchar.h	
LibCtype	ctype.h, wctype.h	
LibTime	time.h	
LibLocale	locale.h	
LibMath	math.h	
LibGdtoa	-- Do Not Use --	This library is used internally and should not need to be explicitly specified by an application. It must be defined as one of the available library classes in the application's DSC file.

These libraries must be fully described in the [LibraryClasses] section of the application package's DSC file. Then, each individual application needs to specify which libraries to link to by specifying the Library Class, from the above table, in the [LibraryClasses] section of the application's INF file. The AppPkg.dsc, AppDemoPkg.dsc, and Enquire.inf files provide good examples of this.

Within the source files of the application, use of the Standard headers and library functions follow standard C programming practices as formalized by the ISO 9899:1990, with Addendum 1, (C 95) C language specification.

IMPLEMENTATION Specific Features

It is very strongly recommended that applications not use the **long** or **unsigned long** types. The size of this type varies between compilers and is one of the less portable aspects of C. Instead, one should use the UEFI defined types whenever possible. Use of these types, listed below for reference, ensures that the declared objects have unambiguous, explicitly declared, sizes and characteristics.

UINT64	**INT64**	**UINT32**	**INT32**	**UINT16**	**CHAR16**
INT16	**BOOLEAN**	**UINT8**	**CHAR8**	**INT8**	
UINTN	**INTN**			**PHYSICALADDRESS**	

There are similar types declared in sys/types.h and related files.

The types UINTN and INTN have the native width of the target processor architecture. This width will not change between compilers. Thus, INTN on IA32 has a native width of 32 bits while INTN on X64 and IPF have native widths of 64 bits.

For maximum portability, data objects intended to hold addresses should be declared with type intptr_t or uintptr_t. These types, declared in sys/stdint.h, can be used to create objects capable of holding pointers. Note that these types will generate different sized objects on different processor

architectures. If a constant size across all processors and compilers is needed, use type PHYSICAL_ADDRESS.

Each application's INF file must include a [BuildOptions] section specifying flags necessary to allow the Intel® UADK headers and libraries to be used instead of the libraries and header files provided by the compiler tool chain vendor. Normally this section will be specified as:

```
[BuildOptions]
  INTEL:*_*_*_CC_FLAGS    = /Qdiag-disable:181,186
   MSFT:*_*_*_CC_FLAGS    = /Od
    GCC:*_*_*_CC_FLAGS    = -O0 -Wno-unused-variable
```

Descriptions of the Library Classes comprising the C Standard Library must be included in your application package's DSC file. The application package's DSC file also must include some specialized directives. Two libraries from EDK II must be built specially in order to allow the C library to link with them. These libraries are specified in the [Components] portion of the DSC file as:

```
[LibraryClasses]
  #
  # Common Libraries
  #
  BaseLib|MdePkg/Library/BaseLib/BaseLib.inf
  BaseMemoryLib|MdePkg/Library/BaseMemoryLib/BaseMemoryLib.inf
  #
  # C Standard Libraries
  #
  LibC|AppPkg/Library/LibC.inf
  LibStdLib|AppPkg/Library/StdLib/StdLib.inf
  LibString|AppPkg/Library/String/String.inf
  LibWchar|AppPkg/Library/Wchar/Wchar.inf
  LibCType|AppPkg/Library/Ctype/Ctype.inf
  LibTime|AppPkg/Library/Time/Time.inf
  LibStdio|AppPkg/Library/Stdio/Stdio.inf
  LibGdtoa|AppPkg/Library/gdtoa/gdtoa.inf
  LibLocale|AppPkg/Library/Locale/Locale.inf
  LibUefi|AppPkg/Library/Uefi/Uefi.inf
  LibMath|AppPkg/Library/Math/Math.inf
  LibSignal|AppPkg/Library/Signal/Signal.inf
  LibGcc|AppPkg/Library/LibGcc/LibGcc.inf

[Components]
  MdePkg/Library/BaseLib/BaseLib.inf {
    <BuildOptions>
      MSFT:*_*_*_CC_FLAGS = /X /Zc:wchar_t /GL-
  }
  MdePkg/Library/BaseMemoryLib/BaseMemoryLib.inf {
    <BuildOptions>
      MSFT:*_*_*_CC_FLAGS = /X /Zc:wchar_t /GL-
  }
```

These directives will create instances of the BaseLib and BaseMemoryLib library classes that are built with Link-time-Code-Generation disabled. This is necessary when using the Microsoft tool

chains in order to allow the library's functions to be resolved during the second pass of the linker during Link-Time-Code-Generation of the application.

Though not specifically required by the ISO/IEC 9899 standard, this implementation of the *Standard C Library* provides the following system calls which are declared in `sys/EfiSysCall.h`.

close	read	write	unlink
dup2	rmdir	isatty	open
creat	fcntl	mkdir	fstat
lstat	stat	lseek	truncate
ftruncate			

The open function will accept file names of "stdin:", "stdout:", and "stderr:" which cause the respective streams specified in the UEFI System Table to be opened. Normally, these are associated with the console device. When the application is first started, these streams are automatically opened on File Descriptors 0, 1, and 2 respectively.