



Distribution and development environment for ST40 platforms

Data Brief

Description

The ST40 STLinux distribution and development environment provides everything required to build Linux based systems for ST40-based platforms.

The kernel can be ported to customer boards or used directly on ST reference platforms.

STLinux is open source, delivered in both binary and source form, making it easy to extend and enhance its capabilities for your specific platform.

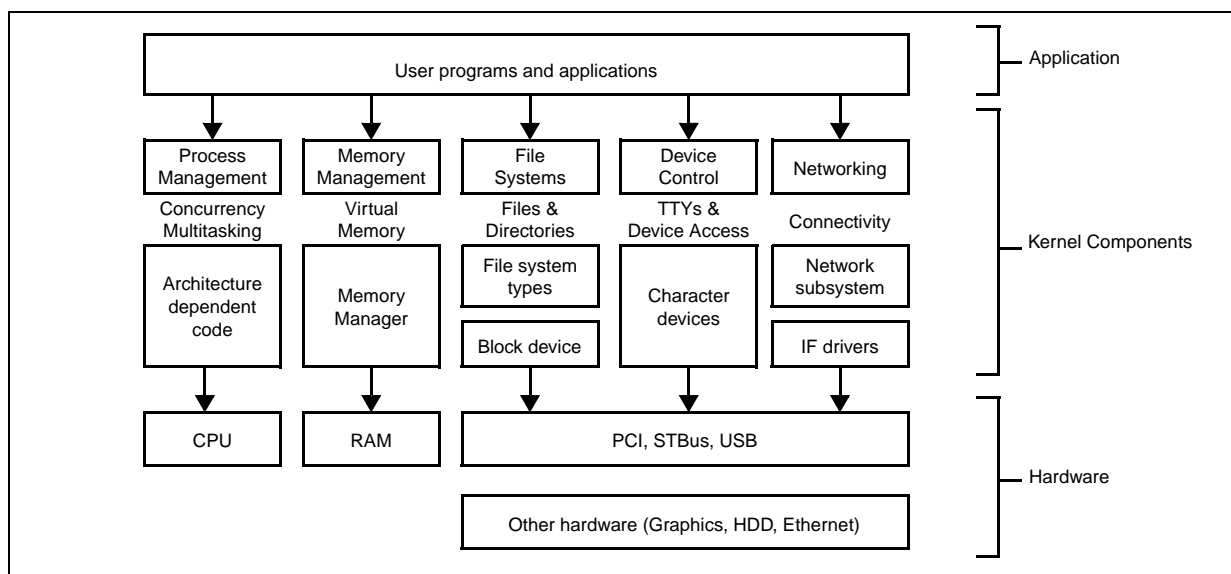
New kernel-space code (such as additional device drivers) can be added. User-space applications, written in ANSI C or C++, can take advantage of the rich application level APIs provided by Linux.

A powerful set of cross development tools make development easy. The tools include state-of-the-art system analysis and trace tools.

During development, STLinux supports a convenient network based paradigm using ST Micro Connect to download the kernel, and using NFS to mount the root file system.

Features

- Open source Linux operating system, tools and development environment based on Linux 2.6 kernel technology, ported and optimized for ST40 based platforms.
- Full set of drivers for basic system devices. STLinux is compatible with higher level driver sets such as the STAPI drivers for A/V.
- Full C and C++ toolsets, based on GNU compiler technology, for both native and cross development from x86 Linux PCs.
- Full root file system with over 600 packages.
- Das U-Boot boot loader for boot from Flash deployment.
- On-line network update management keeps your installation up to date with the latest STLinux releases and updates.
- On-line support and STLinux training courses available
- All STLinux software can be downloaded in binary and source form free of charge from the STLinux web site www.stlinux.com



Contents

1	Introduction	3
1.1	Development environment	4
1.2	Supported hosts	5
1.3	Supported targets	6
2	STLinux kernel	7
3	STLinux tools	8
3.1	Code generation and object file tools	8
3.2	Code analysis tools	8
3.3	Other tools	9
3.4	STWorkbench	10
4	STLinux root file system	11
4.1	File system management	11
5	Das U-Boot boot loader	12
6	STLinux support	13
7	Additional information	14
7.1	System interface	14
7.2	Acknowledgements	14
8	Revision history	15

1 Introduction

ST40 STLinux is the ST40 targeted variant of the STLinux range of Linux Distribution and Development Environments (LDDE). These LDDE systems comprise:

- a Linux kernel based on the open source Linux 2.6 technology
 - Board Support Packages for a wide range of ST40 reference platforms
 - device drivers for on-chip and board level devices
- GNU code development tools for both cross development and native development
 - hosted on x86 Linux PC or natively on target Linux platform
 - GNU C and C++ compilers for ST40 Linux
 - full **glibc** and reduced footprint **uclibc** C run-time libraries, with NPTL support
 - GNU assembler, linker and other binary utilities
 - GNU GDB user application debugger
 - fully kernel aware GNU GDB cross-debugger interfaced to the target through ST Micro Connect 1 or ST Micro Connect 2 using ST TargetPack technology
- dynamic kernel download tool
 - ST Micro Connect 1 or ST Micro Connect 2 based JTAG kernel loader
 - download and run STLinux, passing kernel boot-time parameters from host
 - integrated with kernel-aware GDB debugger to debug downloaded kernel through JTAG
- Das U-Boot boot loader
 - download kernel and file system images to the target over either Ethernet or USB
 - manage on-board NOR and NAND Flash resources for storing environmental data, kernel images, file system images, and so forth
 - boot from Flash, boot over network, and so forth
- advanced system analysis and trace tools
 - based upon dynamic kernel probes inserted at run-time to trace kernel events
 - low-overhead, capable of tracing an STB type system while it performs a normal A/V decode without interfering with application function
- STWorkbench IDE
 - based on the Eclipse IDE and compatible with STWorkbench products for other ST toolsets.
 - powerful source navigation and editing tools
 - fully integrated with STLinux debug and trace tools
 - ST40 STLinux specific plug-ins to customize the IDE for configuring, building, downloading, debugging and managing STLinux systems
- Linux root file system
 - populated with over 600 ST40 packages to provide a rich workstation like development environment of tools, libraries, utilities and applications.
 - hosted either over NFS or using a variety of file system types on IDE or SATA HDD, USB mass storage, NOR or NAND Flash, and others
 - provided in both **glibc** and **uclibc** based variants
 - BusyBox based shell and basic utilities for low footprint deployment systems

- flexible delivery and support model
 - all software is open source and delivered in both pre-built binary and source form
 - software delivery managed as large number of individual RPM packages, and both binary and source RPMs are provided automatically
 - **stmyum** network update and installation tool to create and installation and maintain it up to date with ongoing STLinux development under user control
 - on-line documentation and support databases

1.1 Development environment

The STLinux development environment is based upon an x86 architecture Linux PC as the development host.

The primary download and debug connection between the host and the target platform is the JTAG port present on all platforms. This connection uses the ST Micro Connect host target interface. Both the ST Micro Connect 1 and the ST Micro Connect 2 are supported.^(a) The ST Micro Connect provides a networked solution to allow multiple development hosts to access a target platform over a LAN.

The ST Micro Connect is connected to the host with an Ethernet network connection and to the target with the JTAG port (using the STMC I/O Converter Type A) for boards with a standard 20-pin JTAG connector, or directly using the LVDS cable for boards supporting the LVDS connector.

The **ST40load_gdb** tool loads, boots and debugs a kernel on a target platform using the ST Micro Connect. The **sh4-linux-gdb** tool can be used directly to debug a kernel already loaded on a target platform.

When booted, the Linux kernel requires a root file system to provide access to the various user space libraries, applications, utilities and tools needed for its operation.

The STLinux kernel can be configured to acquire its root file system from a number of locations, such as:

- a RAM based file system, loaded together with the kernel through the ST Micro Connect
- a Flash based file system burned into the Flash memory on the board
- a disk based file system on an externally attached IDE or SATA hard disk drive, or a form of USB mass storage device
- a network based file system mounted using NFS

This last approach is the most flexible during development, because the development host PC can act as the networked fileservers, exporting part of its own file system to the target to become the target's root file system. The contents of the target root file system can then be manipulated easily and directly from the host PC. For this approach, the STLinux kernel must have its networking support configured and a suitable network connection must exist between the target and the host PC network.

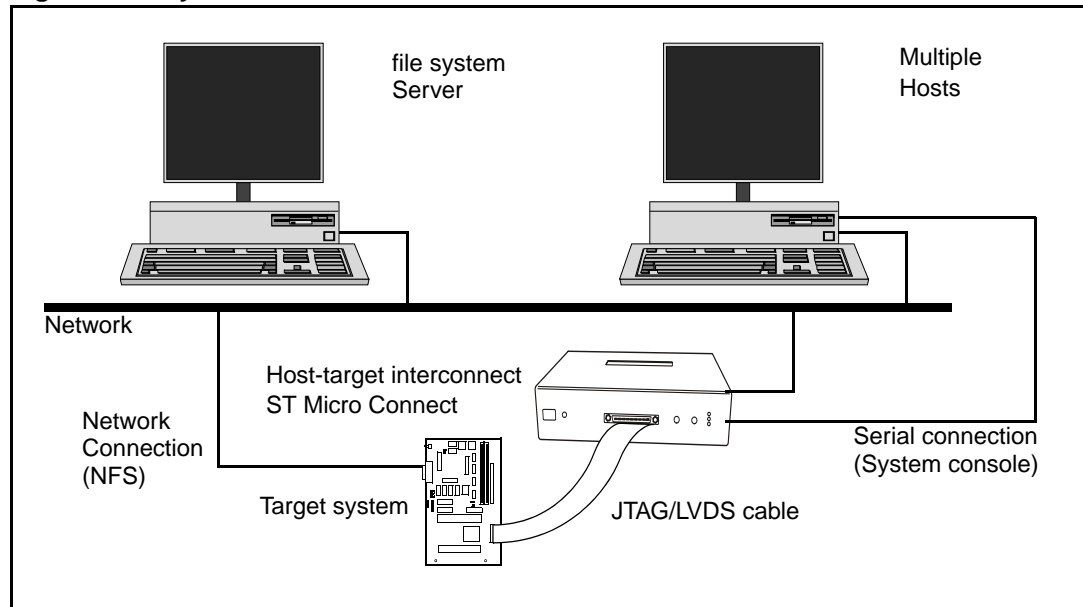
Although not essential, it is often convenient to be able to interact with the target's system console from the host system. This is achieved by configuring the STLinux kernel to support the system console by using a serial port on the target platform. A standard null-modem

a. The first ST Micro Connect product was named the "ST Micro Connect". It is now known as the "ST Micro Connect 1" and the term "ST Micro Connect" is used to refer to the family of ST Micro Connect devices.

(cross over) serial cable connects the serial port to one of the serial connections on the host PC. Any host terminal emulation software (for example, Minicom) can then be used to interact with the STLinux system console directly from the host PC.

Figure 1 shows how the various target connections are used in a system.

Figure 1. System overview



In addition to supporting the dynamic download of kernels and root file systems through the ST Micro Connect, the STLinux distribution contains a version of Das U-Boot boot loader. The boot loader itself can be downloaded to the target platform through the ST Micro Connect and burnt into Flash memory on the board so that it runs automatically on platform reset.

Das U-Boot supports the hardware bring-up of the board and can download other images over the serial or ethernet target connections shown in *Figure 1*. Das U-Boot manages the transfer of these images, such as STLinux kernels, root file system images, and so forth to Flash and on reset can be configured to load the STLinux kernel image to RAM and pass control to it with suitable parameters such that the entire system boots seamlessly and automatically from a board level reset.

1.2 Supported hosts

The STLinux development environment is based upon an x86 architecture Linux PC as the development host.

The distribution and tools are supported on:

- Red Hat Enterprise Linux workstation 3
- Red Hat free Fedora distributions

In practice, many current x86 Linux PC distributions are usable.

1.3 Supported targets

A wide variety of STMicroelectronics target reference platforms are supported with pre-built binary kernels as part of the distribution. For example:

- STi7100/7109 evaluation platform
- STi7109-REF reference platform
- STi7109E-REF reference platform
- STi7200-MBoard reference platform
- STi7111-MBoard reference platform
- STi7105-MBoard reference platform
- STi7141-MBoard reference platform
- STi5202-Mboard reference platform
- STb7109 HD reference platform

In addition, some selected customer designed boards are supported directly.

The kernel is intended to be customizable by the user, and therefore the complete source code is provided, including the standard Linux kernel build scripts. A user can easily modify the kernel configuration using the kernel configuration interfaces and rebuild target support for these platforms with new configurations of kernel features and options. For example, this could be done to remove support for unused peripherals or add support for new ones.

With all the kernel sources provided, it is easy for more experienced users to port the kernel to entirely new platforms, such as customer designed boards that use ST40 based SoCs. All platform dependencies are isolated in platform support source files, making it straight forward to implement a new Board Support Package for a new platform and build a new STLinux kernel to support it.

Expert Linux users can modify and extend the kernel itself to meet their own specific needs.

2 STLinux kernel

The current STLinux kernel is a port of open source Linux kernel 2.6 to the ST40 architecture, with board support packages for a range of STMicroelectronics and customer platforms. At the time of writing, the kernel is based on the kernel.org 2.6.23 release. STMicroelectronics strives to track the evolution of the open source technology as closely as is pragmatically possible, so the minor version number may change frequently.

STMicroelectronics maintains, enhances and optimizes the kernel from kernel.org specifically for the ST40 processor and ST SoCs based upon it.

STMicroelectronics provides device drivers for generic IP on the ST40 SoCs and on the target boards supported.

The kernel is delivered in multiple formats within the distribution.

- As a source RPM, which contains the original kernel.org tar-ball and a series of patches containing the STMicroelectronics enhancements and optimizations for ST40 STLinux. This enables the relationship of ST40 STLinux to the baseline kernel.org tree to be understood.
- As a binary RPM, which contains the fully patched kernel source tree in a form suitable for immediate configuration and building for any supported platform. This enables users to modify, configure and build the kernel themselves using the STLinux tree as their baseline.
- As a series of binary RPMs, which contain pre-built kernels for all supported platforms. These enable any of the supported platforms to be bootstrapped with STLinux OS rapidly and reliably without requiring users to build their own kernels from scratch.

The kernel is supplied with a sophisticated kernel configuration system (**Kconfig**) and kernel build system (**Kbuild**) to largely automate the process of specifying the kernel configuration parameters (using tools such as **menuconfig** and **Xconfig**) and building the configured resident kernel image (**vmlinux**) and any related modules specified in the kernel configuration. The kernel build system can also be used to build any additional modules implemented by the user against the appropriate kernel version and interfaces.

The kernel provides an abstract execution environment for application code, supporting memory management, process management (processes, threads and interrupt abstractions, synchronization primitives, and so forth), file systems, networking, abstractions of character, block device control, interfacing, and so forth.

The Linux kernel supports a full virtual memory model in which user-space applications operate in their own independent virtual address spaces, and presents an efficient, standard kernel API to user-space applications.

STLinux has been optimized for ST40 and for use in embedded systems and supports fast boot, power management, native POSIX threads, fast timers, and so forth.

The diagram on the front cover of this Data Brief shows the overall structure of the kernel in abstract terms.

3 STLinux tools

3.1 Code generation and object file tools

STLinux supports both native and cross code generation tools (assembler, linker, C and C++ compilers, binary utilities, and so forth).

Two C run-time environments are supported:

- GNU C library (**glibc**) providing the full power of the GNU C run-time standard
- **uclibc** for small footprint systems providing an optimized implementation of all major C run-time interfaces

Note: The Native POSIX Thread Library (NPTL) is supported in both the **uclibc** and **glibc** environments and most utilities and packages supplied are built against both run-times.

The major tools components supported are:

sh4-linux-cpp	GNU C/C++ preprocessor
sh4-linux-gcc	GNU C compiler
sh4-linux-g++	GNU C++ compiler
sh4-linux-as	GNU assembler
sh4-linux-ar	Create, modify and extract files from archive file
sh4-linux-ld	GNU Linker
sh4-linux-ranlib	Generate index to archive file
sh4-linux-readelf	Displays information about ELF files
sh4-linux-size	List section sizes and total size in object files
sh4-linux-strings	Display strings of printable characters from object files
sh4-linux-strip	Discard symbols from object files
sh4-linux-nm	List symbols from object files
sh4-linux-objcopy	Copy and translate object files
sh4-linux-objdump	Display information from object files

3.2 Code analysis tools

STLinux provides full GDB based user-space debugging, in both native and cross mode (using **gdbserver**) together with a range of standard Linux trace and profiling tools to enable user-space application analysis and debug.

Additional kernel debug support is provided by **Kgdb** and by the STLinux kernel JTAG debugger, which operates through the ST Micro Connect and uses the target's JTAG port to probe CPU resources and memory. The kernel JTAG debugger is fully aware of kernel data-structures and memory management, which means that it can debug the kernel itself as though it were another user-space application, including code in kernel modules executing in virtual memory spaces. The advantage of the STLinux kernel JTAG debugger over **Kgdb** is that it is enhanced with many kernel specific operations, enabling it to display key kernel data structures and information to the user, even following a kernel crash.

STLinux supports the **KPTrace** kernel event tracing tools, which enables the user to dynamically insert and monitor trace points within the kernel. Any arbitrary kernel symbol can be monitored at low overhead and without rebuilding, or even stopping, the kernel. At each event, information such as the time of occurrence and the stack back-trace can be recorded and later analyzed graphically to examine the dynamic behavior of the complete system. Trace points within an interrupt context are supported. Flight-recorder mode is available, in which the most recent period of activity is stored in a circular buffer and can be examined preceding an event of interest. This mode works even after a kernel crash since the circular buffer is preserved and can be accessed using the JTAG port.

Additional system analysis tools are currently in development for STLinux.

The major tools components currently supported are:

sh4-linux-gdb	GNU target debugger and front end to kernel
Kgdb	GNU GDB front end to ST Micro Connect based kernel debugger
st40load_gdb	Kernel download and launch tool
sh4-linux-gcov	GNU coverage tool
sh4-linux-gprof	GNU profiling tool
oprofile	System-wide profiler for Linux system
kptrace	Kernel event tracing tool

3.3 Other tools

Many other tools are supported as part of the STLinux distribution. There are over 600 different packages that can be installed, many of which provide tools and utilities intended to run on the target system. The packages are too numerous to list here, but many will be familiar to any experienced Linux user. A representative sample is listed below.

System tools

- utilities to create file systems of various formats (**mkfs.jffs2**, **mkfs.xfs**, **mkfs.ext2**)
- utilities to manage Flash memory (**flash_erase**, **flash_info**, **flash_lock**, and others)
- configuration utilities (**initdconfig**, **shellconfig**, and others)
- specific interface utilities (**i2cdetect**, **i2cset**, and others)
- and many others

Standard Linux user utilities

- searching utilities (**find**, **grep**, and so forth)
- process management utilities (**cron**, **top**, **ps**, **kill** and so forth)
- editing utilities (**vi**, **sed**, **awk**, **head**, **tail**, and so forth)
- data processing utilities (**sort**, **col**, **gzip**, **bzip2**, **cpio**, and so forth)
- comparison utilities (**diff**, **cmp**, and so forth)
- file system utilities (**ls**, **mkdir**, **chmod**, **chown**, **cp**, **mv**, **cat**, and so forth)
- and many others

Standard system administration utilities

- user and group management (**groupadd**, **useradd**, and so forth)
- file system management (**mount**, **umount**)
- network utilities (**ping**, **arp**, **ethtool**, **netperf**, **netstat**, and so forth)
- name resolution utilities (**ypset**, **ypbind**, **named**, and so forth)
- and many others

3.4 STWorkbench

STWorkbench is built on the Eclipse IDE. The framework is extended using the CDT (C/C++ Development Tooling) and target specific plug-ins to enable the user to develop, execute and debug STLinux applications interactively.

STWorkbench allows the user to import existing kernel build trees seamlessly into the STWorkbench environment without disruption to the complex Linux KBuild system.

The STWorkbench environment uses the services of the running target directly to support host based browsing and manipulation of the root file system, including the dynamic installation and removal of kernel modules and user space applications built within STWorkbench.

For development, STWorkbench supports the debug of the kernel itself, kernel modules and user space applications using the appropriate debug tools through a common and powerful GUI interface. STWorkbench also supports the profiling of the system with **kptrace**, including powerful data visualization tools to help the user understand the trace data captured.

See the *STWorkbench Data Brief* (ADCS 8063672) for more information.

4 STLinux root file system

The STLinux kernel is the executive core of the system, and responsible for process and memory management, synchronization of concurrent activity, file systems, networking, device abstraction, and so forth. However, most of what the user thinks of as an “operating system” is not the kernel, but the many commands, utilities, libraries, shells, and other tools that reside in the system root file system. As soon as the initialization of the kernel reaches a point where it can run user-space code, control is passed to software within the root file system.

The nature, extent and sophistication of the software in the root file system determines the character of the final system. Consequently, to provide an efficient product, the developer selects only the software that meets the specific needs of the system and discards the remainder.

The opposite philosophy applies during development, when the developer benefits from access to an extensive root file system. The STLinux approach is to provide a well populated root file system for development, rich in tools and utilities. This enables the developer to build and add additional software (including software acquired from the open source community) and cross-build this software using the tools provided.

Using NFS to mount the root file system from the host gives the developer a rich development environment on the target that is similar in scope to a full Linux workstation. When the developed system is ready for deployment, the file system can be stripped down to its bare essentials. At this stage, many standard utilities may be replaced with the low-footprint BusyBox equivalent version; or **glibc** based utilities replaced with **uclibc** equivalents. The file system can then be implemented as a RAM or Flash based file system directly on the target.

4.1 File system management

The STLinux distribution is provided in the form of Red Hat Package Manager (RPM) packages. Each package represents a specific capability (tool, library, daemon, and so forth) and can be downloaded individually from the STLinux website (www.stlinux.com). Currently, STLinux provides over 600 RPMs, giving users the ability to customize the contents of their root file system.

The STLinux website also supports the Yellowdog updater, modified (YUM). This utility helps users keep their installation up-to-date with updated or new packages as they become available by supporting network installation and update of RPM packages.

All RPM packages are available for anonymous ftp from the STLinux website and provide software in both binary and source code format. Binaries can be rebuilt from the sources by using the **rpmbuild** tool to read the .spec files contained in the source RPMs. This strategy avoids the need for complex cross configuration and build processes.

5 Das U-Boot boot loader

The STLinux distribution includes a port of Das U-Boot, a standard open source universal boot loader provided by Denx software. The purpose of Das U-Boot is to cold-boot an STLinux based system from reset.

Like the kernel itself, Das U-Boot is delivered in several forms within the distribution.

- As a source RPM containing the original **u-boot** tar-ball and a series of patches providing ST platform support.
- As a binary RPM containing the fully patched **u-boot** source tree in a form suitable for immediate configuration and building on any supported platform, or porting to a new platform as required.
- As a series of binary RPMs, which contain pre-built **u-boot** boot loaders for all supported platforms.

Das U-Boot is copied onto Flash memory at the reset address of the CPU. When the board is reset, the boot loader gains control and initializes the hardware, loads the kernel (and optionally a file system) from Flash into RAM for execution. Das U-Boot can pass a user-specified set of command line parameters to the kernel.

During development, Das U-Boot can be downloaded and run using the ST Micro Connect. It can load kernels from serial, ethernet, HDD, USB mass storage or Flash. Das U-Boot manages the platform Flash and can burn itself, the kernel and the kernel's root file system into Flash conveniently.

6 STLinux support

Support for the STLinux distribution is provided by the Bugzilla support database, hosted on the bugzilla.stlinux.com website. Users can search the database of existing support requests for relevant information, or send their own specific support request directly to the STLinux development team. We aim to respond quickly and to resolve most requests in a timely manner.

Before submitting a support request, please read the documentation and other information provided on the main STLinux website at www.stlinux.com, as it is likely that the information you require has already been provided.

7 Additional information

The minimum host system requirements for STLinux are as follows:

- 1GHz processor
- 512 Mbyte RAM
- 1024 x 768 High Color screen
- a supported Linux x86 distribution (see [Section 1.2 on page 5](#))

Note: In order to run STWorkbench, the Sun Java Runtime Environment 5.0 or later is required. No other JVMs are supported, and using earlier versions will prevent STWorkbench working correctly.

7.1 System interface

The toolset operates in conjunction with:

- ST Micro Connect 1 (The ST Micro Connect 1 is a legacy product that can no longer be ordered.)
- ST Micro Connect 2 (Order Code: stmc2-20/40/200)

7.2 Acknowledgements

Linux[®] is a registered trademark of Linus Torvalds.

Eclipse[®] is a registered trademark of the Eclipse Foundation

Red Hat[®] is a registered trademark and RPM[™] and Fedora[™] are trademarks of Red Hat Software, Inc.

8 Revision history

Table 1. Document revision history

Date	Revision	Changes
4-Sep-2008	A	Initial release.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2008 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com