UM2210
User manual

Getting started with the X-CUBE-OUT1 firmware package on X-NUCLEO-OUT01A1 for STM32Cube

Introduction

The X-CUBE-OUT1 package lets you drive the ISO8200BQ insulated intelligent power switch to manage any kind of load for industrial applications.

It is an STM32Cube expansion package developed to run on specific STM32 Nucleo boards plus the X-NUCLEO-OUT01A1 development board.

The STM32Cube framework eases portability across different STM32 MCU families.

Contents

List of tables

List of figures

1 Acronyms and abbreviations

Table 1: List of acronyms

Acronym	Description
IPS	Intelligent power switch
GPIO	General purpose input output

2 What is STM32Cube?

STMCube™ represents the STMicroelectronics initiative to make developers' lives easier by reducing development effort, time and cost. STM32Cube covers the STM32 portfolio.

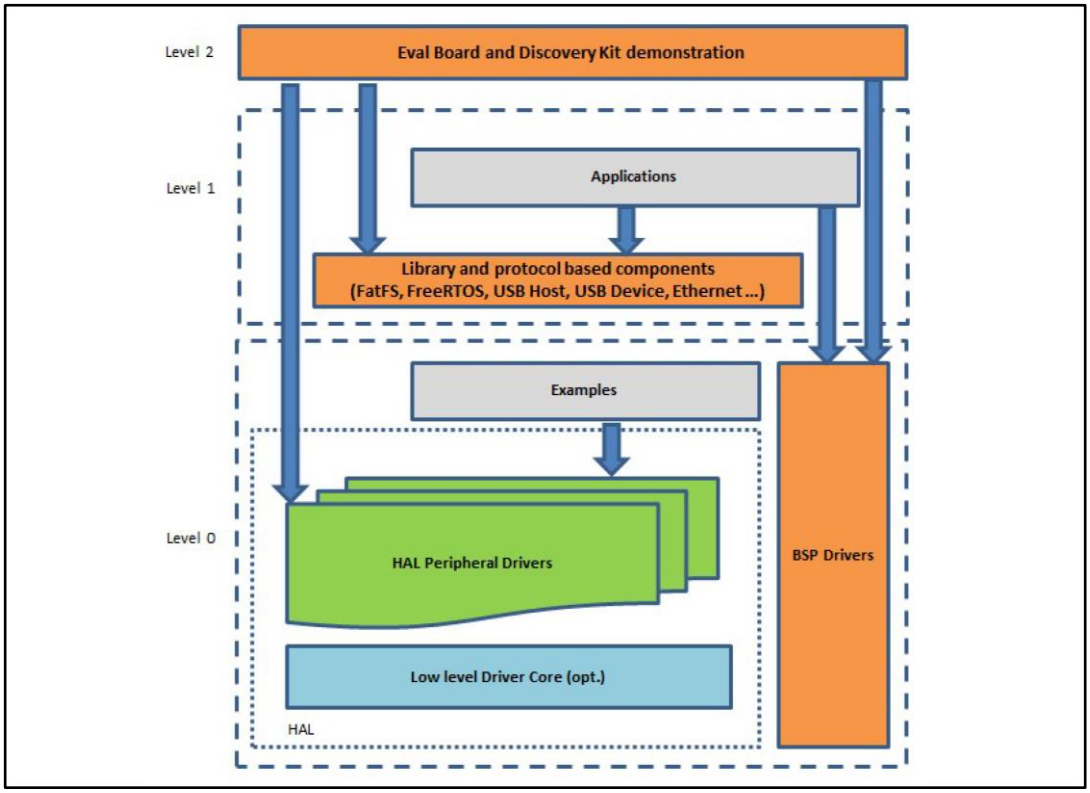
STM32Cube version 1.x includes:

- STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive embedded software platform specific to each series (such as the STM32CubeF4 for the STM32F4 series), which includes:
  - the STM32Cube HAL embedded abstraction-layer software, ensuring maximized portability across the STM32 portfolio
  - a consistent set of middleware components such as RTOS, USB, TCP/IP and graphics
  - all embedded software utilities with a full set of examples

2.1 STM32Cube architecture

The STM32Cube firmware solution is built around three independent levels that can easily interact with one another, as described in the diagram below.

Figure 1: Firmware architecture



Level 0: This level is divided into three sub-layers:

- Board Support Package (BSP): this layer offers a set of APIs relative to the hardware components in the hardware boards (Audio codec, IO expander, Touchscreen, SRAM driver, LCD drivers. etc…); it is based on modular architecture allowing it to be easily

ported on any hardware by just implementing the low level routines. It is composed of two parts:

  - Component: is the driver relative to the external device on the board and not related to the STM32, the component driver provides specific APIs to the external components of the BSP driver, and can be ported on any other board.
  - BSP driver: links the component driver to a specific board and provides a set of easy to use APIs. The API naming convention is BSP_FUNCT_Action(): e.g., BSP_LED_Init(), BSP_LED_On().

- Hardware Abstraction Layer (HAL): this layer provides the low level drivers and the hardware interfacing methods to interact with the upper layers (application, libraries and stacks). It provides generic, multi-instance and function-oriented APIs to help offload user application development time by providing ready to use processes. For example, for the communication peripherals (I²C, UART, etc.) it provides APIs for peripheral initialization and configuration, data transfer management based on polling, interrupt or DMA processes, and communication error management. The HAL Drivers APIs are split in two categories: generic APIs providing common, generic functions to all the STM32 series and extension APIs which provide special, customized functions for a specific family or a specific part number.
- Basic peripheral usage examples: this layer houses the examples built around the STM32 peripherals using the HAL and BSP resources only.

Level 1: This level is divided into two sub-layers:

- Middleware components: set of libraries covering USB Host and Device Libraries, STemWin, FreeRTOS, FatFS, LwIP, and PolarSSL. Horizontal interaction among the components in this layer is performed directly by calling the feature APIs, while vertical interaction with low-level drivers is managed by specific callbacks and static macros implemented in the library system call interface. For example, FatFs implements the disk I/O driver to access a microSD drive or USB Mass Storage Class.
- Examples based on the middleware components: each middleware component comes with one or more examples (or applications) showing how to use it. Integration examples that use several middleware components are provided as well.

Level 2: This level is a single layer with a global, real-time and graphical demonstration based on the middleware service layer, the low level abstraction layer and basic peripheral usage applications for board-based functions.

3 X-CUBE-OUT1 firmware expansion for STM32Cube

3.1 Overview

The X-CUBE-OUT1 firmware package expands the functionality of STM32Cube software.

The package features:

- Software package to build industrial digital output applications based on the ISO8200BQ device
- Easy portability across different MCU families thanks to STM32Cube
- Sample applications included for hands-on practice
- Driver layer for easy ISO8200BQ device management
- Free user-friendly license terms
- Implemented on a NUCLEO-F103RB, NUCLEO-F302R8 or NUCLEO-F401RE development board with mounted X-NUCLEO-OUT01A1 expansion board

All the sample implementations can be selected by pushing the blue user button on the STM32Nucleo development board.

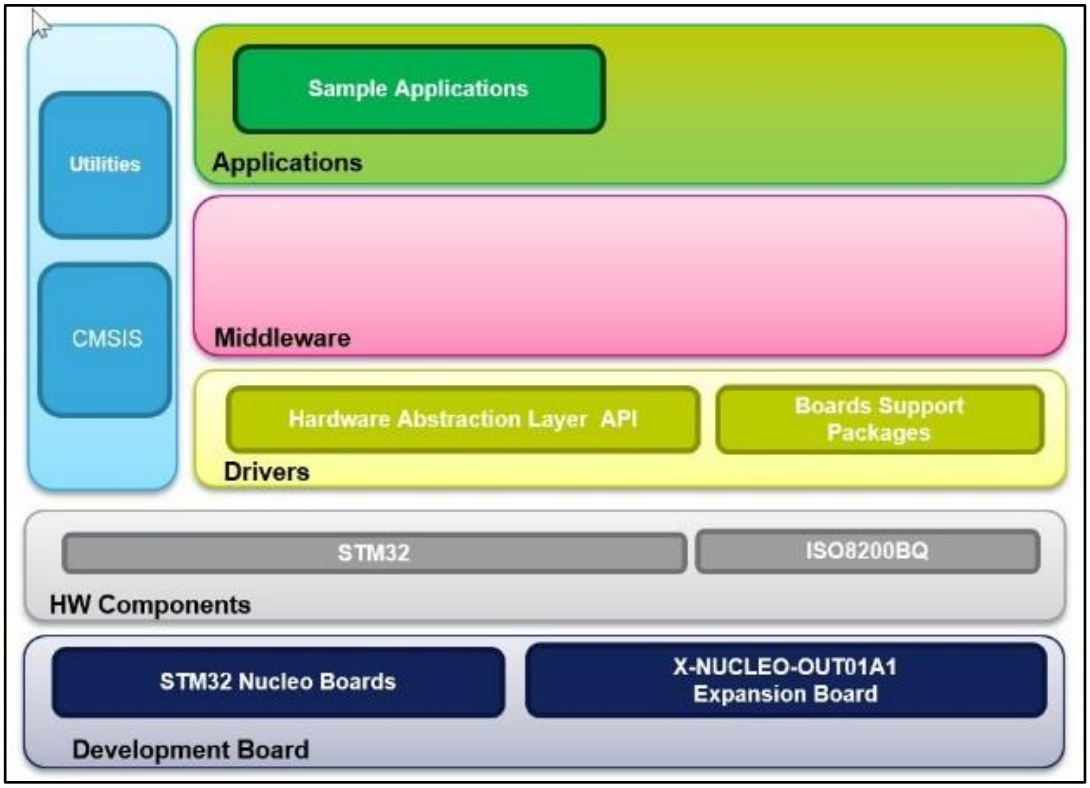
3.2 Architecture

This firmware is developed in compliance with STM32Cube architecture and is intended for the development of applications using the drivers for industrial automation. It is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller and extends STM32Cube with a specific Board Support Package (BSP).

The firmware layers used by the application to access and use the driver are:

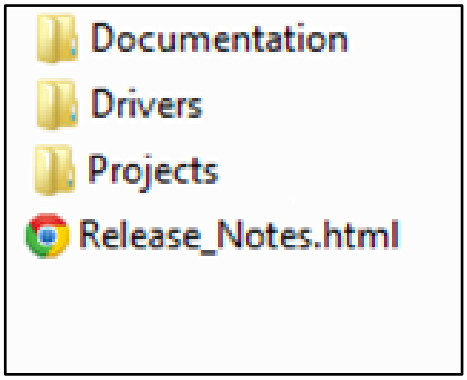
- STM32Cube HAL layer: a simple, generic and multi-instance set of application programming interfaces (APIs) to interact with the upper layers (application, libraries and stacks). Its generic architecture allows other layers built on it (like middleware) to function without specific microcontroller unit (MCU) configuration data. This improves library code reusability and portability across other devices.
- Board Support Package (BSP) layer: supports the peripherals (except MCU) on the STM32 Nucleo board. This limited set of APIs provides a programming interface for timer and GPIO management.
- Middleware layer: not implemented in this solution.

Figure 2: X-CUBE-OUT1 firmware architecture



3.3 Folder structure

Figure 3: X-CUBE-OUT1 folder structure



The folders include:

- **Documentation**: with a compiled HTML file generated from the source code, detailing the software components and APIs.
- **Drivers**: contains the HAL drivers, the board specific drivers (BSP) for each supported board or hardware platform (including on-board components) and the CMSIS vendor-independent hardware abstraction layer for the ARM® Cortex®-M processor series.
- **Projects**: with sample applications to drive the output stage:
  - compiled for: NUCLEO-F103RB, NUCLEO-F302R8 and NUCLEO-F401RE platforms
  - development environments: IAR Embedded Workbench for ARM, Keil Development Tool, SW4STM32

3.4 APIs

Detailed API function and parameter descriptions are available in a compiled HTML file in the package Documentation folder.

3.5 Firmware description

The firmware is conceptually divided into different levels, which improves code flexibility and adaptability to different hardware setups.

3.5.1 Top level firmware

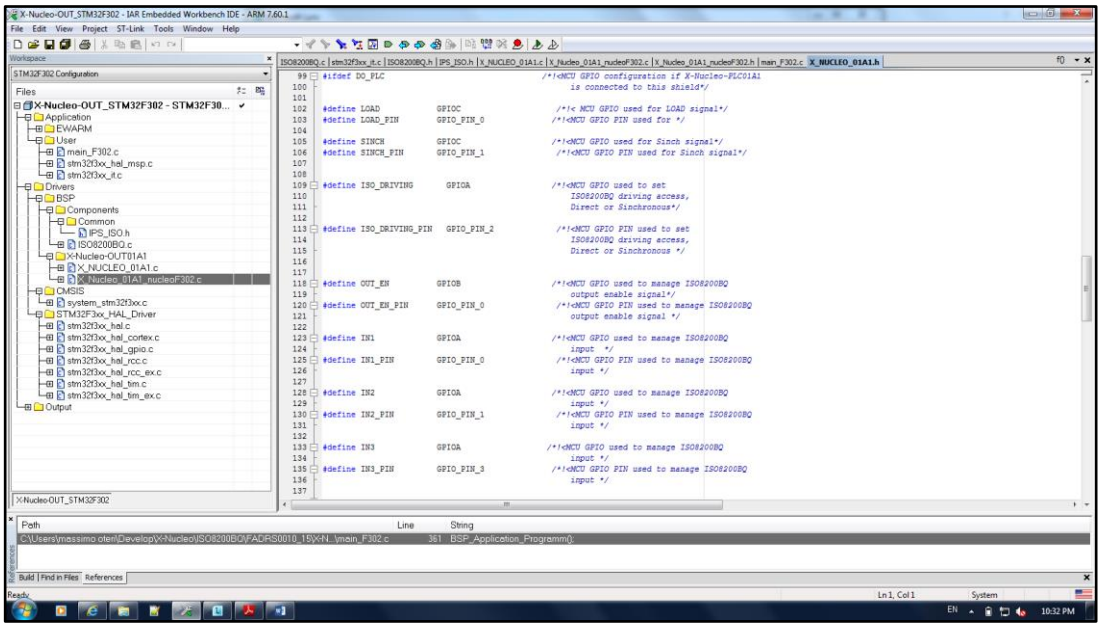
The top level firmware is associated with the physical connection between the STM32Nucleo and X-NUCLEO-OUT01A1 boards.

Its implementation is given in the following X-NUCLEO-OUT01A1 files:

- **.h**: gives the correspondence between the STM32 GPIOs and the Arduino™ connectors
- **.c**: lists the routines used to drive the resources available on the Arduino connector

To change the GPIO resources, you only need to modify the #define in *X-NUCLEO-OUT01A.h* accordingly.

Figure 4: GPIO pin defines in X-NUCLEO-OUT01A1.h



Looking at the above figure, to change the GPIO correspondence for the pin IN1, you modify #define IN1 with the new GPIO port name and #define IN1_PIN with the new GPIO pin number.

3.5.2 Low level firmware

The low-level implementation is associated with the direct driving of the ISO8200BQ IC, given in the ISO8200BQ **.c** and **.h** files. In these files, the user can find a direct relation between the routine used and the IC pin.

3.6 Firmware examples

Different examples have been developed for you to get started with tool evaluation. The main routines are written in *X-NUCLEO-01A1.c*:

- `BSP_AllPin_Low()`: this routine permits the driving mode (Direct or Synchronous) to simultaneously switch-OFF all 8 output channels.
- `BSP _AllPin_High()`: this routine permits the driving mode (Direct or Synchronous) to simultaneously switch-ON all 8 output channels
- `BSP _DrivePin_GROUP(uint8_t channel)`: this routine permits the driving mode (Direct or Synchronous), to simultaneously change the output channel states in group mode.
- `BSP _DrivePin_SINGLE(uint8_t channel, uint8_t Status)`: this routine enables the driving mode (Direct or Synchronous). To change the single output channel state, put the number of channels involved and the new state (1 → ON, 0 → OFF).

The above routines linked with Timer interrupt can toggle the output status for a fixed period. The default period is 1 second, but this can be changed via the `Frequency Required` variable in the *X-X_Nucleo_01A1_nucleoFxx.h* file[a].

You can also modify the duty cycle by changing the `Duty` variable in the same file.

[a] this file is copied in the X-NUCLEO-OUT01A1 folder and it is different for each STM32Nucleo board

4 System setup guide

4.1 Hardware description

4.1.1 STM32 Nucleo platform

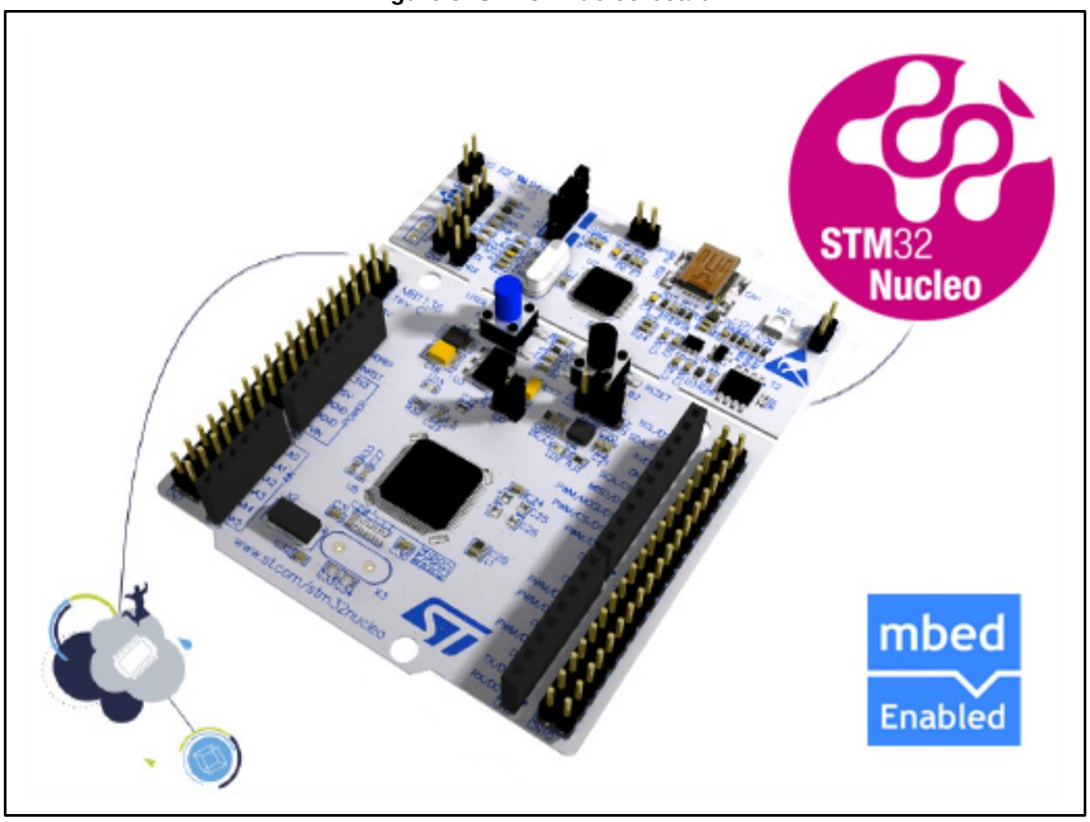
STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

The Arduino™ connectivity support and ST morpho connectors make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide range of specialized expansion boards to choose from.

The STM32 Nucleo board does not require separate probes as it integrates the ST-LINK/V2-1 debugger/programmer.

The STM32 Nucleo board comes with the comprehensive STM32 software HAL library together with various packaged software examples.

Figure 5: STM32 Nucleo board



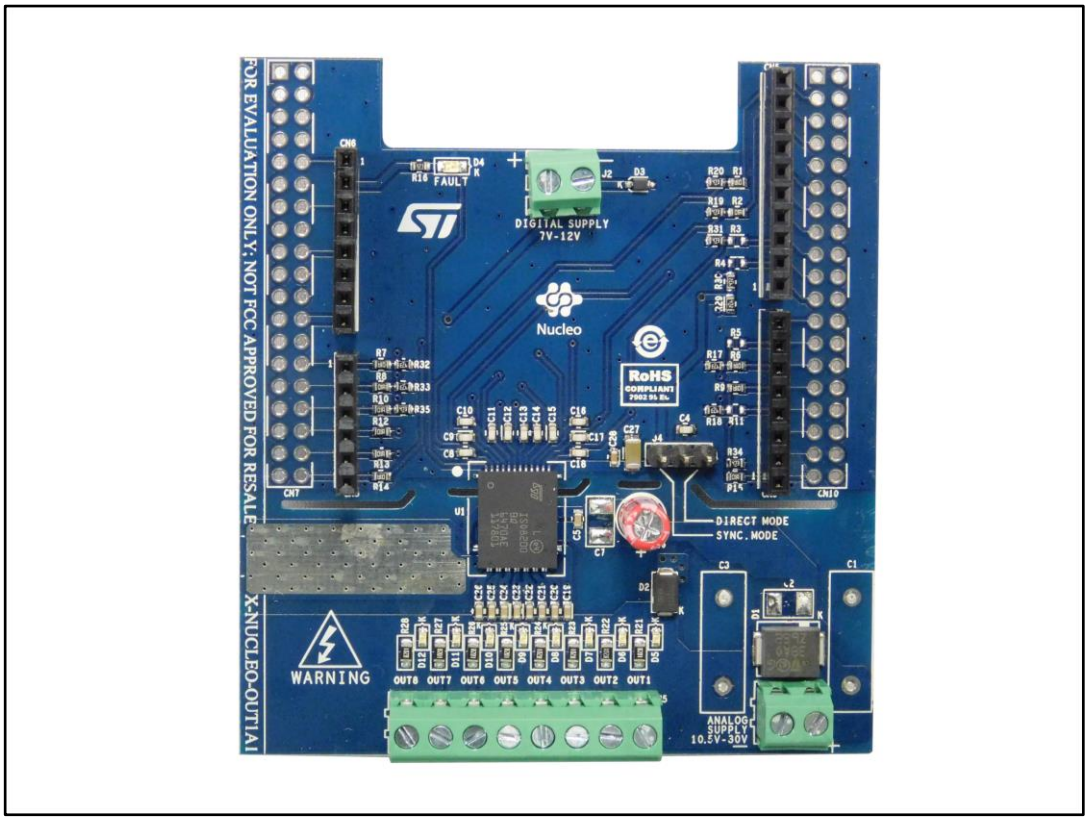
Information regarding the STM32 Nucleo board is available at *www.st.com/stm32nucleo*

4.1.2 X-NUCLEO-OUT01A1 expansion board

The X-NUCLEO-OUT01A1 expansion board has Arduino™ UNO R3 connector layout compatibility and can be used with the STM32 Nucleo development board. It is designed around the ISO8200BQ intelligent power switch (IPS) by ST, with integrated insulation.

The device can be handled by the STM32 MCU in parallel mode, using only the GPIOs and the MCU timer in case the developer needs to enable output switching.

Figure 6: X-NUCLEO-OUT01A1 expansion board



Information regarding the X-NUCLEO-OUT01A1 expansion board is available on *http://www.st.com/x-nucleo*

4.2 Software description

The following software components are needed to set up a suitable development environment:

- X-CUBE-OUT1: STM32Cube software expansion pack for industrial application development, available on www.st.com.
- A development tool-chain and compiler supported by the STM32Cube framework:
  - IAR Embedded Workbench for ARM® (EWARM) toolchain + ST-LINK
  - Keil µVision5 Development Tool (MDK-ARM) toolchain + ST-LINK
  - SW4STM32 Development Tool + ST-LINK

5 References

The following reference material is freely available on www.st.com

- ISO8200BQ datasheet
- X-NUCLEO-OUT01A1 user manual

6 Revision history

Table 2: Document revision history

Date	Version	Changes
07-Jun-2017	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.