MICROCHIP

AN3761

KSZ DSA Driver Utilization

Author: *Jacky Hou*
Microchip Technology Inc.

INTRODUCTION

This application note describes the KSZ Distributed Switch Architecture (DSA) subsystem and how to use the KSZ DSA drivers for a switch application. The design of DSA is to use unmodified network tools such as *bridge*, *ip*, and *ifconfig* to configure or query a KSZ switch port.

The examples and the software demonstrated or introduced in this document are based on Microchip EVB-KSZ9477 evaluation board running the Microchip GitHub software.

The KSZ DSA supported part numbers are:

- KSZ8463 and equivalents
- KSZ9477 and equivalents
- KSZ8895 and equivalents
- KSZ8863 and equivalents
- KSZ8795 and equivalents

Note:	Users should have a basic understanding of Switchdev.

Sections

This document includes the following topics:

References

Consult the following documents for details on the specific parts referred to in this document.

- *KSZ9477S Data Sheet* (www.microchip.com/DS00002392)
- EVB-KSZ9477 Software and User Guides (https://github.com/Microchip-Ethernet/EVB-KSZ9477)
- Linux® Kernel Documentation

Terms and Abbreviations

The following terms and abbreviations are used in this document:

- CPU port – the port facing the host processor and is referred to as the CPU port in DSA terminology
- DSA – Distributed Switch Architecture
- Front ports – the switch user ports and is referred to as the secondary device interfaces in DSA terminology
- MDIO/SPI/I^{2}C controller – the primary device role in the control path
- MDIO/SPI/I^{2}C device – the secondary device role in the control path
- NIC/MAC – Network Interface Controller, the primary device interface in DSA terminology

In addition, the following terms are interchangeable in this document:

- *frame* and *packet*
- *front port* and *user port*
- *external CPU* and *host processor*

DSA ARCHITECTURE

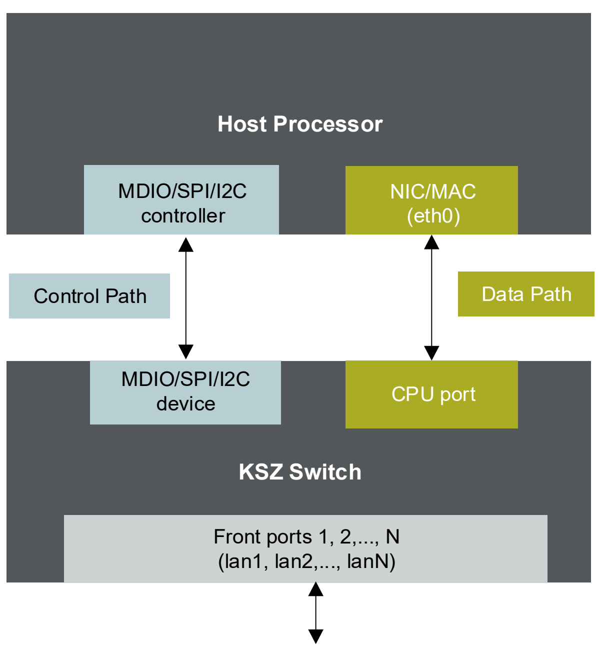
A KSZ switch is typically comprised of multiple front ports, and one CPU port connecting to an Ethernet controller of an external CPU, which is for receiving frames from the switch or sending frames to the switch.

For each front port, KSZ DSA creates a network device that is used as a controlling and frame-flowing endpoint, which is also used by the Linux networking stack. These network interfaces are referred to as secondary device network interfaces in the DSA terminology and code. (The “eth0” in a Linux network application is the primary device interface). Specifically, each device works as an independent Network Interface Controller (NIC).

The idea for using KSZ DSA is that the KSZ switch supports a proprietary Tail Tagging mode, which is a hardware feature making the switch insert the tail tagging bytes at the end of each frame it received to and from specific ports. That is, Tail Tagging provides the ingress and egress port information between the host processor and the switch.

Figure 1 is a simplified block diagram showing a typical KSZ DSA driver enabled application (for example, EVB-KSZ9477), including control and data paths.

FIGURE 1: KSZ DSA BLOCK DIAGRAM



Note:	Switch ports are modeled as Linux network interfaces (secondary or primary), and DSA does not create secondary network devices for the CPU port.

KSZ SWITCH TAGGING PROTOCOL

As mentioned, KSZ DSA supports tagging protocols by inserting the tail tagging bytes at the end of the packets. The frame format of the tail tag is shown in Figure 2.

FIGURE 2: KSZ TAIL TAG FRAME FORMAT

BYTES	6	6	(4)	2	46 (42) – 1500	1 or 2 (From Host) 1 (To Host)	4
	DEST ADDRESS	SOURCE ADDRESS	802.1Q TAG	ETYPE or LENGTH	PAYLOAD	TAIL TAG	FCS

When the switch forwards a received packet to the host port, the switch adds one tail tagging byte to the packet to indicate to the host processor which port the packet is received from.

In reverse, the host processor must add one or two tail tagging bytes to each packet that it sends to the switch to indicate the intended egress ports. When multiple priority queues are enabled, Tail Tagging is also used to indicate the priority queue. Tail Tagging applies only to the CPU port other than the front ports.

Note: As the host processor should handle the tail tagging bytes by the DSA driver, the packet forwarding throughput between the CPU port and the host processor is lesser as compared to a regular driver. For IPerf test on EVB-KSZ9477, it is 40% lower.

NETWORK DEVICES

In DSA driver implementation, two kinds of network devices are required: the primary device and the secondary device.

- **Primary device**

  Primary network devices are unmodified network device drivers (that is, reuse the original working network device drivers) for the CPU port of the switch. Such devices act as a pipe between the host processor (the external CPU) and the KSZ switch. A typical name is **eth0**.

- **Secondary device**

  Secondary network devices created by DSA are stacked on top of the primary device. Each of these network devices is responsible for being a controlling and frame-flowing endpoint for each front port.

  Secondary device interfaces are used for inserting or removing the switch tagging protocol when forwarding frame between the switch ports, and they are also used for querying the switch for *ethtool* operations, such as port statistics, port link speed/duplex mode, and so on.

  A typical name is **lan0**, **lan1**,…, **lan(N-1)**, **lanN** for front ports 1, 2, …, N, N+1.

Data path in the transmit and receive directions is as follows:

- Transmit direction (from host processor to switch CPU port):

  secondary device interface > tagger routine > primary device interface > cpu port > remove tag > front ports

- Receive direction (from switch front ports to host processor via switch CPU port):

  front port > add tag > cpu port > netif_receive_skb() in primary device interface > tag parser routine > netif_receive_skb() in secondary device interface

UTILIZING SWITCHDEV

As mentioned, for control in DSA, Switchdev is used to interface with the bridge layer, including VLAN filtering through the Linux network stack's control plane. Therefore, in Linux kernel configuration, CONFIG_SWITCHDEV must be enabled, so that the DSA framework can leverage the Switchdev files, `net/switchdev`.

Note: Switchdev is not a driver model and is not involved in the data plane but the control plane only. For the data plane, DSA behaves like a port connected to an external CPU's NIC or MAC port to form the plane.

SNIPPET OF KERNEL BOOT MESSAGE

To check if the KSZ DSA driver is loaded, the kernel booting message can be monitored and checked if the following or a similar message is printed:

```
Starting kernel ...
…
…
                       / * truncated the intermediate message */
…
…
DSA: switch 0 0 parsed
DSA: tree 0 parsed
libphy: dsa slave smi: probed
Microchip KSZ9477 dsa-0.0:00: attached PHY driver [Microchip KSZ9477] (mii_bus:phy_addr=dsa-
0.0:00, irq=-1)
Microchip KSZ9477 dsa-0.0:01: attached PHY driver [Microchip KSZ9477] (mii_bus:phy_addr=dsa-
0.0:01, irq=-1)
Microchip KSZ9477 dsa-0.0:02: attached PHY driver [Microchip KSZ9477] (mii_bus:phy_addr=dsa-
0.0:02, irq=-1)
Microchip KSZ9477 dsa-0.0:03: attached PHY driver [Microchip KSZ9477] (mii_bus:phy_addr=dsa-
0.0:03, irq=-1)
Microchip KSZ9477 dsa-0.0:04: attached PHY driver [Microchip KSZ9477] (mii_bus:phy_addr=dsa-
0.0:04, irq=-1)
```

KSZ DSA LIMITATIONS

The KSZ DSA driver does not support the following features:

- IEEE1588/PTP message indication
- Egress queue priority change
- KSZ switches cascading
- Multiple Tagging ports

KSZ DSA DRIVER AND DEMONSTRATION SOFTWARE

The driver in Microchip GitHub (https://github.com/Microchip-Ethernet) is maintained by Microchip, and the driver is usually not the same as the version in Linux mainline that is maintained by Linux Open Source community. Users should reference the Microchip GitHub code when debugging DSA issues.

Source Files

Driver: `<kernel base>/drivers/net/dsa/microchip/*`

It is required to implement the dsa_switch_ops structure to support DSA-based switch configuration, PHY management, bridge layer functions (VLAN and STP), and port statistics.

For example, in `drivers/net/dsa/microchip/ksz9477.c`, the `ksz9477_switch_ops` implements the switch operations, such as:

- `ksz_port_bridge_join()` and `ksz_port_bridge_leave()` that are for bridge management
- `ksz9477_port_vlan_filtering()`, `ksz_port_vlan_prepare()`, `ksz9477_port_vlan_add()`, and `ksz9477_port_vlan_del()` that are for VLAN management

The `ksz9477_tag_ops` is used in the DSA protocol layer, which retrieves the tail tagging bytes (see Figure 2).

To check the KSZ part numbers supporting DSA, search the following data structures in the folder:

`ksz8463_chip_names[ ]`	`ksz8863_chip_names[ ]`
`ksz9477_chip_names[ ]`	`ksz8795_chip_names[ ]`
`ksz8895_chip_names[ ]`	

Protocol: `<kernel base>/net/dsa/tag_ksz.c`

It is required to register CPU port tagging operations (that is, the Tail Tag handler).

For example, in `net/dsa/tag_ksz.c`, the `ksz_xmit()` and `ksz_rcv()` operates on the `sk_buff` and inserts or parses the tail tagging bytes extracted by the DSA driver.

Device Tree Source: `<kernel base>/arch/arm/boot/dts/*.dts`

It is required to define the host processor's device tree describing the peripherals resource to be used.

For example, `at91-sama5d3_xplained_ung8071.dts` is for EVB-KSZ9477.

In `at91-sama5d3_xplained_ung8071.dts`, the following snippet of user ports and CPU port bindings can be seen. The `spi1: spi@f8008000` node describes how the user ports and the CPU ports are defined via the name = "value" properties.

For user ports, it is similar to the following:

```
port@0 {
     reg = <0>;
     label = "lan1";
 };
```

The above can be copied and used the same way as in the DTS file.

For CPU ports, it is similar to the following:

```
port@5 {
    reg = <5>;
    label = "cpu";
    ethernet =
            <&macb0>;
    fixed-link {
        speed = <1000>;
        full-duplex;
    };
};
```

The interface is defined as fixed link speed and duplex mode via the "fixed-link" subnode with 1000 Mbps link speed and full-duplex setting because it is not an MDIO-managed PHY device, and thus cannot perform auto-negotiation process to negotiate the link mode.

Also, note that `phy-mode = "rgmii-txid"` is defined for the CPU port's interface, which is an RGMII interface connected to the SAMA5D36A processor on the EVB-KSZ9477.

GitHub Software Build

1. Use the EVB-KSZ9477 GitHub code (v1.2.1 is not supported to build the KSZ DSA source code; use later releases).
2. Follow the build instruction (see *EVB-KSZ9477 Source Build Instructions Guide*) using DSA-enabled kernel configuration, `sama5_ksz_dsa_defconfig`, in the configuration file, `EVB-KSZ9477/KSZ/Atmel_SOC_SAMA5D3/buildroot/.config`.
   For example, specify the Linux Kernel configuration in the above `.config` file as follows:
   `BR2_LINUX_KERNEL_CUSTOM_CONFIG_FILE="$(KSZ_HOME)/kernels/linux-4.9.143/arch/arm/configs/sama5_ksz_dsa_defconfig"`

Alternatively, build the software using the following:

```
$ git clone https://github.com/Microchip-Ethernet/EVB-KSZ9477.git
$ cd EVB-KSZ9477/KSZ
$ export KSZ_HOME='pwd'
$ cd Atmel_SOC_SAMA5D3/buildroot
$ make O=mybuild atmel_sama5d3_xplained_ksz9897_defconfig
```

At this point, the `mybuild` folder exists. If the `mybuild/.config` file is opened and `KSZ_HOME` is searched, the kernel and the associated `sama5_ksz_defconfig` config file are located.

The application is built with:

```
$ cd mybuild
$ make
```


DEMONSTRATIONS

This section demonstrates the use of Linux networking tools to configure the KSZ switch, such as *ip* for interfaces, *bridge* for bridging, and *ethtool* for obtaining port statistics.

Using *ip* to Create a Bridge Device and List Created Devices

```
#ip link add name br1 type bridge /* add a bridge device, br1 */
#ip link show    /* below showing the network devices created with EVB-KSZ9477 */
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default
qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN mode DEFAULT group default qlen 10
    link/can
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group
default qlen 1000
    link/ether 00:10:a1:94:77:10 brd ff:ff:ff:ff:ff:ff
4: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN mode DEFAULT group default qlen 1
    link/sit 0.0.0.0 brd 0.0.0.0
5: lan1@eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue master br0 switchid
00000000 state LOWERLAYERDOWN mode DEFAULT group default qlen 1000
    link/ether 00:10:a1:94:77:10 brd ff:ff:ff:ff:ff:ff
6: lan2@eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue master br0 switchid
00000000 state LOWERLAYERDOWN mode DEFAULT group default qlen 1000
    link/ether 00:10:a1:94:77:10 brd ff:ff:ff:ff:ff:ff
7: lan3@eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue master br0 switchid
00000000 state LOWERLAYERDOWN mode DEFAULT group default qlen 1000
    link/ether 00:10:a1:94:77:10 brd ff:ff:ff:ff:ff:ff
8: lan4@eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue master br0 switchid
00000000 state LOWERLAYERDOWN mode DEFAULT group default qlen 1000
    link/ether 00:10:a1:94:77:10 brd ff:ff:ff:ff:ff:ff
9: lan5@eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue master br0 switchid
00000000 state LOWERLAYERDOWN mode DEFAULT group default qlen 1000
    link/ether 00:10:a1:94:77:10 brd ff:ff:ff:ff:ff:ff
10: lan6@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br0 switchid
00000000 state UP mode DEFAULT group default qlen 1000
    link/ether 00:10:a1:94:77:10 brd ff:ff:ff:ff:ff:ff
11: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group
default qlen 1000
```

Using *bridge* to Add/Delete VLAN Interfaces

```
#bridge vlan add dev lan1 vid 1 pvid untagged
#bridge vlan del dev lan1 vid 1
```

Using *ethtool* to Get lan5 Interfaces Statistics

```
# ethtool -S lan1
```

NIC statistics:

```
tx_packets: 6
     …
     / * truncated the intermediate dump */
     …
     tx_discards: 0
```

Note:	The *sysfs* system, `/sys/class/net`, still works for switch management.

VERIFYING TRAFFIC BETWEEN FRONT PORT AND CPU PORT

The following steps verify the network connection between the CPU port and the external CPU's NIC/MAC port.

1. Set the network devices online. (Note that the primary device must be set up prior to the secondary devices).

   `#ip link set eth0 up`

   `#ip link set lan5 up`

2. Create a bridge, `br0`, and then add `lan5` to it.

   `#ip link add name br0 type bridge`

   `#ip link set dev lan5 master br0`

3. Assign an IP address to `br0` and get them online.

   `#ip addr add 10.9.52.100/24 dev br0`

   `#ip link set dev br0 up`

4. Connect a PC to `lan5` by assigning a static IP address 10.9.52.109.
5. Test (ping) between `lan5` and `br0`. The test should work.

Now, packets forwarding between the CPU port and the host processor is ready.

Additionally, the *tcpdump* utility can be used to monitor the packets and the DSA tag.

```
# tcpdump -x -i eth0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
08:46:09.517381 IP 10.9.52.109 > 10.9.52.100: ICMP echo request, id 1, seq 863, length 40
0x0000:  4500 003c bee6 0000 8001 fef7 0a09 346d
0x0010:  0a09 3464 0800 49fc 0001 035f 6162 6364
0x0020:  6566 6768 696a 6b6c 6d6e 6f70 7172 7374
0x0030:  7576 7761 6263 6465 6667 6869 04

08:46:09.518323 IP 10.9.52.100 > 10.9.52.109: ICMP echo reply, id 1, seq 863, length 40
0x0000:  4500 003c 33d3 0000 4001 ca0b 0a09 3464
0x0010:  0a09 346d 0000 51fc 0001 035f 6162 6364
0x0020:  6566 6768 696a 6b6c 6d6e 6f70 7172 7374
0x0030:  7576 7761 6263 6465 6667 6869 0010
```

In the above captured packets, it can be seen that one tail tagging byte, **`04`**, is inserted at the Rx direction (lan5 to CPU port), and two tail tagging bytes, **`0010`**, are inserted at the opposite (Tx) direction

APPENDIX A: APPLICATION NOTE REVISION HISTORY

TABLE A-1: REVISION HISTORY

Revision Level & Date	Section/Figure/Entry	Correction
DS00003761A (12-17-20)	Initial release	

NOTES:

THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://microchip.com/support

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

ISBN: 978-1-5224-7378-7

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820