KwikNet®

MCF5475 FEC Device Driver

User's Guide

Version 3

First Printing: **January 15, 2005**
Last Printing: **September 15, 2005**

Manual Order Number: PN513-9C

KADAK Products Ltd.
206 - 1847 West Broadway Avenue
Vancouver, BC, Canada, V6J 1Y5
Phone: (604) 734-2796
Fax: (604) 734-8114

TECHNICAL SUPPORT

KADAK Products Ltd. is committed to technical support for its software products. Our programs are designed to be easily incorporated in your systems and every effort has been made to eliminate errors.

Engineering Change Notices (ECNs) are provided periodically to repair faults or to improve performance. You will automatically receive these updates during the product's initial support period. For technical support beyond the initial period, you must purchase a Technical Support Subscription. Contact KADAK for details. Please keep us informed of the primary user in your company to whom update notices and other pertinent information should be directed.

Should you require direct technical assistance in your use of this KADAK software product, engineering support is available by telephone, fax or e-mail. KADAK reserves the right to charge for technical support services which it deems to be beyond the normal scope of technical support.

We would be pleased to receive your comments and suggestions concerning this product and its documentation. Your feedback helps in the continuing product evolution.

KADAK Products Ltd.
206 - 1847 West Broadway Avenue
Vancouver, BC, Canada, V6J 1Y5

Phone: (604) 734-2796
Fax: (604) 734-8114
e-mail: amxtech@kadak.com

DISCLAIMER

KADAK Products Ltd. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability and fitness for any particular purpose. Further, KADAK Products Ltd. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of KADAK Products Ltd. to notify any person of such revision or changes.

TRADEMARKS

AMX in the stylized form and KwikNet are registered trademarks of KADAK Products Ltd.
AMX, AMX/FS, InSight, *KwikLook* and KwikPeg are trademarks of KADAK Products Ltd.
Microsoft, MS-DOS and Windows are registered trademarks of Microsoft Corporation.
All other trademarked names are the property of their respective owners.

KwikNet MCF5475 FEC Device Driver User's Guide

Table of Contents

This page left blank intentionally.

KwikNet MCF5475 FEC Device Driver

1. Introduction

The Freescale ColdFire MCF5475 includes a dual channel Fast Ethernet Controller (FEC) that supports 10Mbps and 100Mbps data transfer rates. The FEC can operate only in conjunction with the MCF5475 Multi-channel DMA (MCDMA) controller that provides DMA services to a number of MCF5475 subsystems.

The MCDMA is driven by its private tasks which execute from microcode located in the MCF5475 static RAM memory. These tasks are not to be confused with tasks managed by an RTOS such as KADAK's AMX kernel. Within this document, the term task will be assumed to reference an MCDMA task unless otherwise specified.

The MCDMA API and microcode modules are supplied by Freescale Semiconductor, Inc. You must refer to the following Freescale documents for guidance in the proper setup and use of the MCDMA controller.

MCF548x Preliminary Reference Manual
Multi-channel DMA API User's Guide

Getting Started

The KwikNet MCF5475 FEC Ethernet Device Driver consists of a number of components which collectively support one or two Ethernet network interfaces managed by the KwikNet TCP/IP Stack. These components must be compiled and the resulting object modules must be merged into a library module which will be referred to as the KwikNet MCF5475 MCDMA Library. The device driver can be used with KwikNet CFire (PN513-2) and AMX CFire (PN512-1) or with the KwikNet Porting Kit (PN713-2).

To add the MCF5475 FEC device driver to your application, proceed as follows. Install the driver in its own directory, separate from KwikNet, as described in Chapter 2.

The MCF5475 FEC device driver must be attached to a KwikNet network interface. The network interface is defined as described in Chapter 3. The device driver is then attached to the network interface as described in Chapter 4. The device driver parameters required to configure the driver are specified in the data sheet provided in Appendix A.

Once you have defined the operating characteristics of the network interface and the MCF5475 FEC device driver, you can build your KwikNet Library. You must build the KwikNet Library prior to compiling the driver source modules. By building the KwikNet Library first, you will ensure that all of the required KwikNet header files have been collected together and are ready for your use.

The MCF5475 FEC device driver components must be compiled and the resulting object modules must be merged into the KwikNet MCF5475 MCDMA Library. Your application is then compiled and linked with this library and the KwikNet Library as described in Chapter 5.

2. Installation

The KwikNet MCF5475 FEC Ethernet Device Driver is provided on the KwikNet CD-ROM. The installation process installs the driver files in directory *KNT303\KN5475E* within an installation directory of your choice. Note that this directory is separate from the KwikNet PPC32 installation directory *KNT513* or the KwikNet Porting Kit installation directory *KNT713*.

The driver files are installed in the following subdirectories within installation directory *KNT303\KN5475E*:

ERR	Construction error summary
M5475E	Driver source and header files
MCDAPI	MCDMA API and microcode source and header files
MAKE	Construction make directory
TOOLXXX	Toolset specific files
TOOLXXX\LIB	Toolset specific library will be built here

The KwikNet MCF5475 FEC Ethernet Device Driver consists of the following files:

M5475E\M5475E.H	Device driver header file
M5475E\M5475E.C	Device driver source file
M5475E\M5475EBS.C	Physical device pin assignment and MII setup
M5475E\KN5475MD.C	KwikNet MCDMA initialization source file
M5475E\KN5475MD.H	KwikNet MCDMA header file
MAKE\KN5475E.MAK	KwikNet MCF5475 MCDMA Library make specification file
README.TXT	Version information

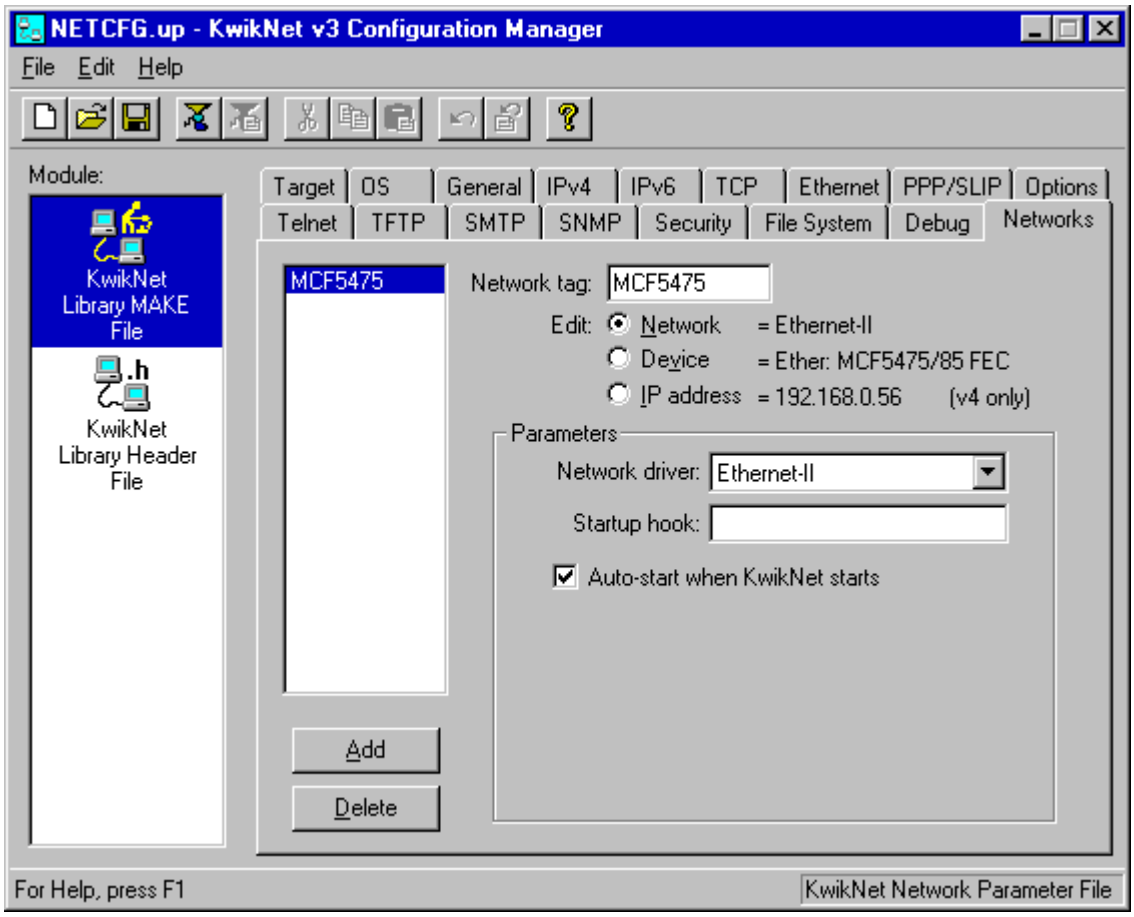
The following MCDMA API and microcode source files are available from Freescale Semiconductor, Inc. These files are provided to you as permitted by the terms of the Freescale source code distribution agreement described in the source files. The files provided by KADAK in directory *MCDAPI* are the most recent variants tested by KADAK. Text file *README.TXT* in the root of the installation directory identifies the MCDMA version and its release date.

MCDAPI\MCD_DMA.H
MCDAPI\MCD_PROGCHECK.H
MCDAPI\MCD_TASKSINIT.H
MCDAPI\ReleaseNotes.txt
MCDAPI\MCD_DMAAPI.C
MCDAPI\MCD_TASKS.C
MCDAPI\MCD_TASKSINIT.C

3. Configuring the Network

You must define each network that your application supports. You can dynamically add a network interface at runtime. Alternatively, you can define your network interface in your KwikNet configuration. Networks defined in this manner will be prebuilt for you when KwikNet is started.

To add a prebuilt MCF5475 Ethernet network interface definition to your KwikNet Library, use the KwikNet Configuration Manager to edit your Network Parameter File. The network parameters are edited on the Networks property page. The layout of the window is shown below. Follow the directions provided in Chapter 2.4 of the KwikNet TCP/IP Stack User's Guide.

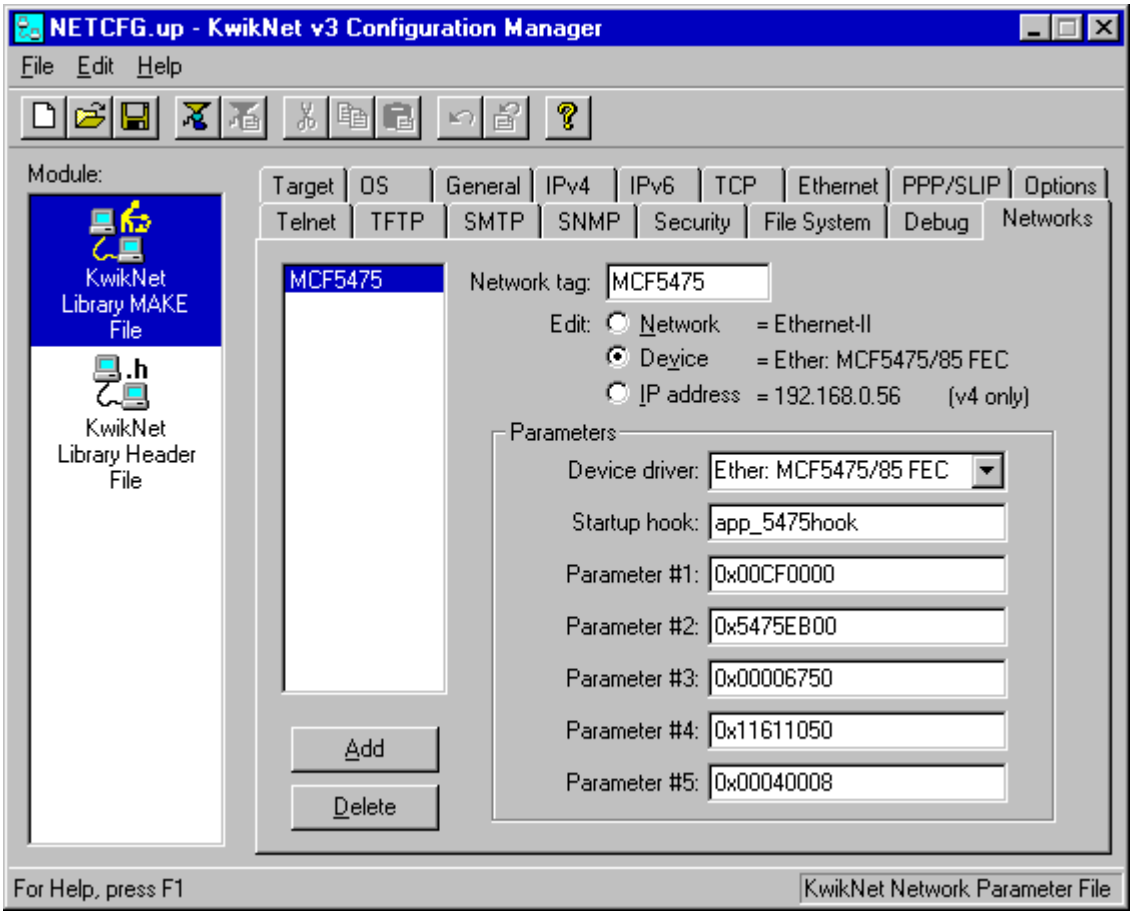


4. Configuring the Device Driver

You must define the KwikNet device driver attached to each network interface that your application supports. If you call KwikNet procedure *kn_ifadd()* to dynamically add a network interface at runtime, you must provide the device driver definition as a parameter in the call.

If you specify a prebuilt network, you must define its device driver by using the KwikNet Configuration Manager to edit your KwikNet Network Parameter File. The driver parameters are edited on the Networks property page. The layout of the window is shown below. Follow the directions provided in Chapter 2.5 of the KwikNet TCP/IP Stack User's Guide.

The numeric device driver parameters used to configure the operating characteristics of the KwikNet MCF5475 FEC Ethernet Device Driver are defined in Appendix A.



5. Making the KwikNet MCF5475 MCDMA Library

The make process depends upon the structure of the KwikNet MCF5475 FEC Ethernet Device Driver installation directory *KN5475E*. When the driver is installed, the following subdirectories are created within directory *KN5475E*.

ERR	Construction error summary
M5475E	Driver source and header files
MCDAPI	MCDMA API and microcode source and header files
MAKE	Construction make directory
TOOLXXX	Toolset specific files
TOOLXXX\LIB	Toolset specific library will be built here

Several toolset specific directories *TOOLXXX* will be present. There will be one such directory for each of the software development toolsets with which KADAK has built the KwikNet MCF5475 MCDMA Library. Each toolset vendor is identified by a unique two or three character mnemonic, *XXX*. The mnemonic *UU* identifies the toolset vendor used with the KwikNet Porting Kit.

Directory	Toolset	Vendor	KwikNet Part
TOOLDA	*DA*	Diab (Wind River)	KwikNet CFire
TOOLME	*ME*	Metrowerks	KwikNet CFire
TOOLUU	*UU*	Custom tools	KwikNet Porting Kit

Within directory *TOOLXXX* you will find a **tailoring file** named *KNZZZCC.INC* used to tailor the KwikNet Library construction process for the Microsoft make utility. This file is a copy of the most recent tailoring file for toolset *XXX* provided with KwikNet CFire . For toolset *UU*, the file is a copy of the most recent Metrowerks tailoring file provided with the KwikNet Porting Kit.

Within directory *TOOLXXX* you will find an additional tailoring file named *KN5475E.INC* used to tailor the library construction process specifically for the generation of the KwikNet MCF5475 MCDMA Library. It is this file which contains the implicit rules for compiling source files and creating the library module.

Getting Ready

Before you start, build your KwikNet Library as described in Chapter 3.2 of the KwikNet TCP/IP Stack User's Guide. Once you have completed this step, all necessary KwikNet header files will be present in your Treck installation directory, say *C:\TRECK\INCLUDE*.

If you are using KwikNet CFire, make sure that your tailoring file *KNZZZCC.INC* from KwikNet installation directory *KNT513\TOOLXXX* matches the copy in driver directory *KN5475E\TOOLXXX*.

If you are using the KwikNet Porting Kit, copy your toolset specific tailoring file *KNZZZCC.INC* from KwikNet Porting Kit working directory *KNT713\TOOLUU* to driver directory *KN5475E\TOOLUU*. Then edit tailoring file *KN5475E\TOOLUU\KN5475E.INC* to use the compilation and librarian commands for your specific software development tools.

Creating the Driver Library

The KwikNet MCF5475 MCDMA Library must be constructed from within the driver installation directory *KN5475E\MAKE*. To create the library, proceed as follows. From the Windows Start menu, choose the MS-DOS Command Prompt from the Programs folder. Make the driver installation *KN5475E\MAKE* directory the current directory.

To use Microsoft's *NMAKE* utility, issue the following command.

```
NMAKE -fKN5475E.MAK "TOOLSET=XXX" "TRKPATH=treckpath"
```

The make symbol *TOOLSET* is defined to be *XXX*, the toolset mnemonic which identifies the software tools which you are using.

The symbol *TRKPATH* is defined to be the string *treckpath*, the full path (or the path relative to directory *KN5475E\MAKE*) to your Turbo Treck TCP/IP installation directory.

For example, assume that the Turbo Treck TCP/IP release from Treck Inc. has been installed in directory *C:\TRECK*. Then, to build the driver library using Microsoft's *NMAKE* utility and Metrowerks tools, issue the following command.

```
NMAKE -fKN5475E.MAK "TOOLSET=ME" "TRKPATH=C:\TRECK"
```

During the library construction process, all KwikNet device driver source files and MCDMA source files will be compiled and the resulting object modules will be placed in directory *KN5475E\TOOLXXX\LIB*. The KwikNet MCF5475 MCDMA Library *KN5475E.A* will be created from these object files and placed in directory *KN5475E\TOOLXXX\LIB*. Note that the library file extension will be *.A* or *.LIB* or some other extension as dictated by the toolset which you are using.

The KwikNet MCF5475 MCDMA Library file *KN5475E.A* must be linked with your application. Revise your KwikNet application link specification to include this library after the KwikNet Library.

> Note
>
> The MCDMA files listed in directory *MCDAPI* may vary as new versions of the MCDMA software are released. If you receive an updated version of the MCDMA source code directly from Freescale, you may have to edit the library make specification file *KN5475E.MAK* to revise the list of MCDMA source files being compiled and to correct their dependencies.

6. Special Considerations

6.1 MCDMA Initialization

The KwikNet MCF5475 FEC Ethernet Device Driver depends upon the MCDMA API and microcode supplied by Freescale Semiconductor, Inc.

The MCDMA microcode must be loaded into memory and initialized once prior to initialization of any of the subsystems that it supports. Each subsystem, such as the FEC, must then start and assume control of the MCDMA tasks upon which the subsystem depends.

Initialization of the MCDMA firmware presents the system designer with some difficult startup issues. A number of different parts of an application will depend upon the MCDMA firmware. Each part must initialize the MCDMA firmware properly for use by that part. The KwikNet MCF5475 FEC Ethernet Device Driver is one such part. And yet the MCDMA microcode must be loaded and initialized only once.

You, as the system designer, must ultimately bear responsibility for loading and initializing the MCDMA firmware to meet the needs of all parts of your application which depend upon MCDMA services.

An MCDMA initialization procedure *kn_MCDMASetup()* is provided with this device driver in source file *KN5475MD.C*. This procedure must be called once, and only once, by your application as it starts up. It must be called before KwikNet procedure *kn_enter()* is called to launch KwikNet.

In a multitasking system, procedure *kn_MCDMASetup()* must be called from an application task. In a single threaded system, it must be called from your KwikNet App-Task executing in the user domain.

Procedure *kn_MCDMASetup()* handles the relocatable loading of the MCDMA firmware image into system memory. It then initializes the MCDMA API and microcode for subsequent use by the KwikNet MCF5475 FEC Ethernet Device Driver. It then clears all pending MCDMA channel interrupts and installs a KwikNet compatible interrupt handler to service subsequent MCDMA channel interrupts.

You are free to edit source file *KN5475MD.C* to adapt the MCDMA firmware loading and initialization sequence to your particular needs. Alternatively, you can merge the code from module *KN5475MD.C* into your own MCDMA initialization module.

> Note
>
> The MCDMA firmware must be loaded and initialized once, and only once, by your application software.

MCDMA Interrupt Management

The KwikNet MCF5475 FEC device driver provides an interrupt handler to service the sixteen possible interrupts from the MCF5475 Multi-channel DMA controller. Each FEC uses two DMA channels for FEC Ethernet transmission and reception. Hence, if both FECs are used, four DMA channels must be reserved for use by the FEC device driver.

The maximum number of supported channels is specified by symbol *NCHANNELS* which is defined in MCDMA header file *MCD_DMA.H*. The DMA channels are numbered from 0 to *NCHANNELS-1*. Your KwikNet device interface configuration must identify the DMA channels to be used with a particular FEC.

The MCDMA interrupt handler is located in source file *KN5475MD.C*. The handler is installed as soon as your application calls procedure *kn_MCDMASetup()* to initialize the MCF5475 Multi-channel DMA controller. When the MCF5475 FEC device driver is called upon to open its Ethernet interface, it calls procedure *kn_MCDMASetIntHandler()* to install its own handler to service the receive interrupt generated by the DMA channel assigned to the FEC receiver. When the interface is closed, the driver calls the same procedure to remove its DMA channel handler. Note that the DMA channel used by the FEC for transmission does not generate a DMA channel interrupt. The FEC transmit interrupt which signals the end of an Ethernet transmission is handled by the device driver's FEC interrupt handler.

The MCF5475 FEC device driver will use two or four of the sixteen available DMA channels. The remaining DMA channels can be used by your application for other purposes. However, since the MCF5475 Multi-channel DMA controller generates a single processor interrupt for any of its sixteen channels, there can only be a single interrupt service procedure for the MCDMA. Your application must therefore use the MCDMA interrupt handler provided with the MCF5475 FEC device driver.

Your application can call procedure *kn_MCDMASetIntHandler()* to install and remove a DMA channel interrupt handler for a specific DMA channel. The calling sequence is specified in the description of the function in source file *KN5475MD.C*. Your handler executes in the context of the interrupt service routine. Hence, your handler must service the interrupting device and remove the source of the interrupt request. The handler must execute as quickly as possible and must not perform any operations other than those permitted within an interrupt handler by your RTOS or single-threaded OS.

You are free to edit source file *KN5475MD.C* to adapt the MCDMA interrupt handler to your particular needs. Alternatively, you can merge the code from module *KN5475MD.C* into your own MCDMA interrupt handling module, being sure to retain all of the public functions and their functional integrity.

MCF5475 Slice Timer Usage

A custom timing service is implemented in source file *KN5475MD.C* for the benefit of the MCF5475 FEC device driver. Precise timing is derived from slice timer number 0 within the MCF5475 Slice Timer subsystem. Procedure *kn_tbdelay()* can be used to instrument a compute bound delay with microsecond precision. This procedure supercedes the crude delay function offered by KwikNet procedure *kn_brddelay()*. Note that proper operation of procedure *kn_tbdelay()* depends upon the slice time clock frequency as defined by symbol *SLT_FREQ* in header file *KN5475MD.H*.

MCF5475 Register Support

Two low level MCF5475 register access functions are provided in source file *KN5475MD.C*.

Function *kn_get_mbar()* returns the Memory Base Address Register (*MBAR*) value to the caller. Since the MCF5475 *MBAR* cannot be read, the function returns the *MBAR* value defined by symbol *MBAR_ADDRESS* in header file *KN5475MD.H*.

Function *kn_get_slt()* reads the Slice Timer Count Value Register and returns its value to the caller.

MCF5475 System SRAM Use

The MCDMA microcode uses a collection of transmit and receive descriptors to control the transmission and reception of Ethernet frames by the DMA controller. These descriptors must be located in uncached data memory. The MCF5475 FEC device driver reserves a region of the MCF5475 static RAM memory (system SRAM) for this purpose.

The region of system SRAM used by the driver is determined by two definitions in driver header file *KN5475MD.H*. Symbol *FEC_SRAM_BASE* specifies the offset of the reserved region within the 32 Kb bank of system SRAM. Symbol *FEC_SRAM_SIZE* specifies the number of bytes in the reserved region.

The region must be 16-byte aligned. The region size must be a multiple of 16. By default, the last 512 bytes of system SRAM (offset *0xFE00* in SRAM) are reserved for the driver's use. This specification provides storage for 64 descriptors of 8 bytes each.

The driver allocates receive and transmit descriptors for FEC0 at the base of the reserved SRAM region. Receive and transmit descriptors for FEC1 are allocated at the end SRAM region.

You can edit the definitions of *FEC_SRAM_BASE* and *FEC_SRAM_SIZE* in file *KN5475MD.H* to adjust the location and size of the reserved SRAM region. If only one FEC is used, you may wish to shrink the size to preserve SRAM for other uses. If the total number of receive and transmit buffers required for FEC0 and FEC1 exceeds 64, you will have to edit the symbol definitions to increase the SRAM storage reserved for the driver.

If you have an alternate region of uncached data memory which can be used for MCDMA descriptor storage, you can edit function *kn_get_fecsram()* near the end of file *KN5475MD.C* to return a pointer to the memory region reserved for use by FECn.

6.2 MCDMA Source Code

KADAK has successfully used the MCDMA API and firmware source code without modification. Using the Metrowerks software development tools, all modules have been successfully compiled without warnings or errors.

However, when the MCDMA source code is compiled using the Diab (Wind River) tools, a number of warnings are encountered. KADAK has made no attempt to review or repair the source code which generated warnings. However, many years of experience with these tools have proven that it is not wise to ignore the warnings and Freescale should be encouraged to review the source code.

> Note
>
> No improper operation of the MCDMA API and microcode has been observed. Note that the MCDMA software has been exercised using the **FEC subsystem only**.

6.3 Configuring the Physical Device

The Ethernet transceiver device (PHY) is configured by function *kn_5475_pcon_fec()* in the driver's pin assignment and board setup module *M5475EBS.C*. The PHY is configured each time the device driver is called by the KwikNet Ethernet network driver to open the network interface for use by the application.

The PHY wiring mode is determined by an attribute setting in the device driver parameter list (see Appendix A). If attribute bit W is 0, the 18-wire (MII) mode is assumed. If attribute bit W is 1, the 7-wire mode is assumed.

If the PHY operates in 7-wire mode, there is little special initialization required. The PHY is configured and is immediately available for use.

If the PHY operates in 18-wire mode using the Media Independent Interface (MII) protocol, the PHY is programmed to initiate an auto-negotiation process in accordance with the Ethernet link speed and duplex mode settings in the device driver parameter list. The auto-negotiation can take awhile, during which time the device is unavailable for use. The negotiation can take from a few tens of milliseconds to one or two seconds to complete.

The device driver cannot suspend execution of the KwikNet TCP/IP stack while it awaits the PHY negotiation results. The driver offers three modes of operation to cope with this issue.

Mode 0: No Delay

In the simplest case, the driver ignores the negotiation process and returns immediately to the network driver which declares the interface open for business. However, your application must defer from using the interface for at least two seconds to ensure that the PHY setup has been completed. This deferral may not be possible if, for example, you have enabled DHCP or AutoIP IP address negotiation to begin once the interface is open.

Mode 1: Continuous Poll

If your application can withstand the consequences, the driver can continuously poll the PHY, up to some maximum interval, waiting for the auto-negotiation to complete. The driver will then return to the network driver, the network interface will be declared open and the network will in fact be fully operational.

However, you must be aware that KwikNet will experience a compute-bound delay which can exceed 2 seconds, every time the MCF5475 FEC network interface is opened for use. In a multitasking system, all lower priority activity will be inhibited until the interface is ready or until the timeout interval expires.

If the PHY auto-negotiation fails to complete within the timeout interval, the driver will report the failure and the network driver will declare that the network interface cannot be opened for use.

Mode 2: Periodic Sampling

By default, the MCF5475 FEC device driver is configured to use a third, preferred method of operation. The driver uses a KwikNet timer to periodically sample the PHY, watching for the auto-negotiation to complete. The sampling period, the wait interval and an additional post-negotiation delay are all configurable parameters. By default, the driver samples at the KwikNet clock frequency for up to 3 seconds, with no additional delay.

The device driver reports the success or failure of the auto-negotiation process to the Ethernet network driver. If a link has been established, the network driver completes the opening process and declares the network interface operational. If necessary, IP address negotiation will then be initiated using either DHCP or AutoIP according to your network interface definition.

If the PHY auto-negotiation fails, the Ethernet network driver will declare that the network interface cannot be opened for use.

Choosing the Mode of Operation

To alter the MCF5475 FEC device driver's PHY initialization strategy, you must edit the device driver header file *M5475E.H*. Define symbol *DD_WAITANC* to be *DD_NOWAIT* for no delay, *DD_POLLING* for continuous polling and *DD_SAMPLING* (the default) for periodic sampling. The timeout interval(s) and sampling interval can be defined as described in the file.

Event Callback Notification

When the periodic sampling method is used (*DD_WAITANC* is *DD_SAMPLING*), the device driver generates a network event to signal the success (*KN_EV_DVCUP*) or failure (*KN_EV_DVCFAIL*) of the PHY auto negotiation.

If you implement the KwikNet event notification function *kn_netevent()* as described in Chapter 4.5 of the KwikNet TCP/IP Stack User's Guide, your function will be called whenever either of these network events is generated. Note that your function will be called again when the network is subsequently declared open (up) or closed (down).

Appendix A. MCF5475 FEC Device Driver Data Sheet

The KwikNet Device Driver Technical Reference Manual describes the design and implementation requirements for all KwikNet device drivers. Appendix C of that manual illustrates the layout of a data sheet used to describe each unique KwikNet device driver.

The KwikNet MCF5475 FEC Ethernet Device Driver conforms to the naming conventions and operating specifications presented in the KwikNet Device Driver Technical Reference Manual. The following data sheet specifies the device driver parameters supported by this driver.

This page left blank intentionally.

KwikNet®

Device Driver

MCF5475 FEC Ethernet

KADAK Products Ltd.
206 - 1847 West Broadway Avenue
Vancouver, BC, Canada, V6J 1Y5
Phone: (604) 734-2796
Fax: (604) 734-8114

MCF5475 FEC Ethernet
Device Driver Data Sheet

Filename *M5475E.C*, *M5475E.H*, *M5475EBS.C*

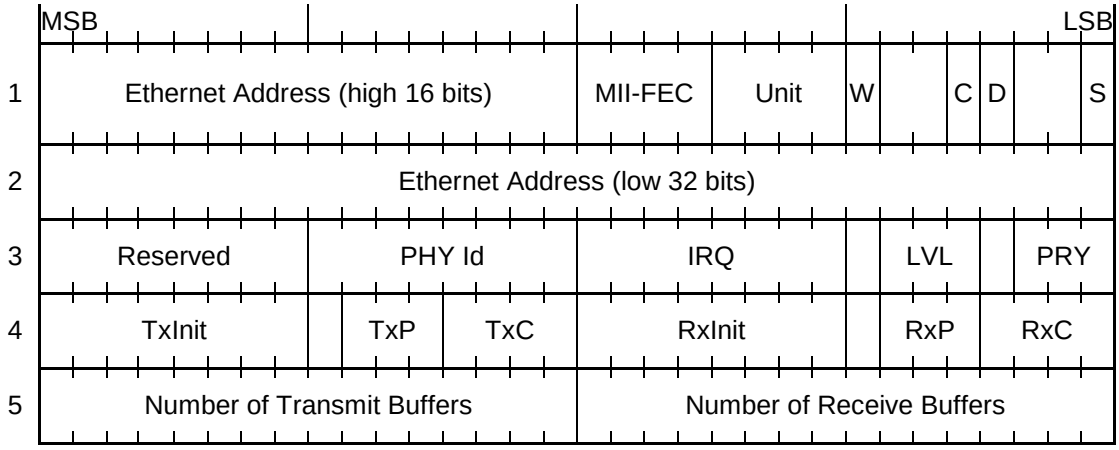
Mnemonic *M5475*

Description The KwikNet MCF5475 FEC Ethernet Device Driver supports the use of the MCF5475 Fast Ethernet Controller (FEC) for 10Mbps or 100Mbps Ethernet communication. The driver depends upon the Multi-channel DMA (MCDMA) controller for transmission and reception of Ethernet frames via the FEC. The driver is provided ready for use on the Freescale MCF5475 Evaluation Board.

Configure This device driver includes a number of low level configurable parameters which can only be adjusted by editing the definitions in the driver source module *M5475E.H*. The default values and editing instructions are included in the file. These parameters control the following features:

- Include support for effects of data caching
- Adjust transmit inactivity timeout
- Select PHY initialization strategy
- Minimum number of transmit and receive buffers
- Support the enforcement of *n*-byte alignment of receive buffers
- Support the enforcement of *n*-byte alignment of transmit buffers

Parameters The KwikNet Configuration Manager can be used to define the following set of device dependent parameters. These parameters are entered in the Device Driver Definition on the Networks property page.



Unit This parameter selects the FEC unit to be used for network communication. The MCF5475 FEC controller supports two Ethernet interfaces, FEC0 and FEC1. Set this parameter to 0 or 1 to select FEC0 or FEC1 respectively for the network interface. Note that the MCF5475 FEC device driver will support the concurrent operation of two Ethernet interfaces with one configured for FEC0 and the other for FEC1.

MII-FEC Each MCF5475 FEC can control its own Media Independent Interface (MII) to configure the physical Ethernet adapter (PHY). In some cases, one of the FECs might control a single MII to manage multiple PHYs. This parameter selects the FEC unit to be used to control the PHY for the Ethernet interface indicated by parameter Unit.

Attributes The low order 8 bits of the first parameter describe the device attributes supported by the driver. Reserved and undefined fields must be 0.

S This bit defines the FEC operation speed. If this bit is 0, the FEC will operate at 10Mbps. If this bit is 1, the FEC will operate at 100Mbps.

D This bit defines the duplex mode of operation. If this bit is 0, the FEC interface will operate in half duplex mode. If this bit is 1, the FEC interface will operate in full duplex mode.

Attributes (continued)

C This bit determines if the driver must accommodate data caching. If this bit is 0, it will be assumed that data caching may be enabled and that caching effects must be considered. If this bit is 1, it will be assumed that data caching will not be present and that caching effects can be ignored.

Note that this attribute will only be effective if symbol `DD_DCACHE` in header file `M5475E.H` is defined to be 1, thereby enabling the device driver to support this data cache accommodation feature.

W This bit determines the physical device (PHY) wiring mode selection. If this bit is 0, the 18-wire (MII) mode is assumed. If this bit is 1, the 7-wire mode is assumed.

Parameters (continued)

Ethernet Address The 48-bit Ethernet address assigned to the network must be defined. The most significant byte (see MSB in diagram) of the high 16 bits of the Ethernet address will be transmitted first; the least significant byte (see LSB in diagram) of the low 32 bits of the address will be transmitted last.

IRQ, LVL, PRY Parameter IRQ is the KwikNet IRQ identifier which defines how the FEC interrupt request is vectored. It is used by the board driver module *KN_B5475.C* to map logical device interrupts to the underlying target hardware and operating system.

Note that in the ColdFire interrupt architecture, interrupt sources are mapped to the processor vector numbers reserved for interrupts. The FEC0 interrupt source is mapped to vector number 103; the FEC1 interrupt source is mapped to vector number 102.

When using KwikNet with AMX CFire, the AMX vector number is used as the KwikNet IRQ identifier. The AMX vector number is simply the processor vector number.

If you are using the KwikNet Porting Kit, the assignment of IRQ identifiers must be done by you as part of your overall system design.

Parameter LVL defines the interrupt request level to be assigned to the FEC interrupt request. Request levels from 1 (lowest) to 6 (highest) are acceptable. Parameter PRY defines the interrupt priority assigned to the FEC interrupt within the request level. Priority values from 0 (lowest) to 7 (highest) are acceptable. The device driver programs the Interrupt Control Register (ICR) for the selected FEC unit to interrupt at priority PRY on request level LVL. The assignment of these parameters must be done by you as part of your overall system design so that all of the required ICRs are programmed with unique and non-overlapping level and priority values. Failure to do so can result in undefined behaviour.

PHY Id This parameter identifies the PHY number (0 to 31) to be presented to the MII when configuring the physical Ethernet adapter. If the MII controller supports only a single physical interface, set this parameter to 0. The MCF5475 Evaluation Board provides a single MII controller for which PHY ids 0 and 1 are to be used to identify the two Ethernet connectors.

Parameters (continued)

RxC, RxP, RxInit Parameter RxC defines the DMA channel (0 to 15) to be assigned to the MCDMA FEC receive task.

Parameter RxP defines the MCDMA FEC receive task priority. The task priority can range from 0 (lowest) to 7 (highest).

Parameter RxInit specifies the initiator to be associated with the MCDMA FEC receive task. Acceptable values are defined in header file *KN5475MD.H*. You can assign receive task initiator *DMA_SRC16*, *DMA_SRC18* or *DMA_SRC22* for use with FEC0. You can assign receive task initiator *DMA_SRC20*, *DMA_SRC24* or *DMA_SRC30* for use with FEC1.

TxC, TxP, TxInit Parameter TxC defines the DMA channel (0 to 15) to be assigned to the MCDMA FEC transmit task.

Parameter TxP defines the MCDMA FEC transmit task priority. The task priority can range from 0 (lowest) to 7 (highest).

Parameter TxInit specifies the initiator to be associated with the MCDMA FEC transmit task. Acceptable values are defined in header file *KN5475MD.H*. You can assign transmit task initiator *DMA_SRC17*, *DMA_SRC19* or *DMA_SRC23* for use with FEC0. You can assign transmit task initiator *DMA_SRC21*, *DMA_SRC25* or *DMA_SRC31* for use with FEC1.

Buffers These parameters define the number of transmit and receive buffers which the MCF5475 FEC device driver must provide for use with the FEC specified by parameter Unit. Unless you edit the definitions in header file *M5475E.H*, you must specify a minimum of one transmit buffer and two receive buffers. Two transmit buffers and eight receive buffers are frequently sufficient.

Descriptors for the transmit and receive buffers will be allocated in Buffer Descriptor Tables located in the uncached MCF5475 system SRAM as described in Chapter 6.1. The driver is configured with enough SRAM to allocate a total of 64 descriptors. If the total number of receive and transmit buffers required for FEC0 and FEC1 exceeds 64, you will have to edit the definitions of *FEC_SRAM_BASE* and *FEC_SRAM_SIZE* in file *KN5475MD.H* to allocate more system SRAM for use by this driver.

Parameters (continued)

The driver's private receive and transmit data buffers are allocated using the standard KwikNet memory allocation services. Each buffer will be at least 1520 bytes in size, rounded up if necessary to meet your alignment and cache line requirements. Your KwikNet memory allocation method (see the OS property page in Chapter 2.3 of the KwikNet TCP/IP Stack User's Guide) must provide enough memory to meet the storage requirements that you have specified.

The MCDMA microcode requires 4-byte alignment of receive and transmit buffers. Unfortunately, since Ethernet headers are 14 bytes long, the data in each received Ethernet frame will be halfword aligned, leading to serious data processing inefficiencies. All receive and transmit buffers must be aligned on an *n*-byte boundary where *n* is defined by symbols *DD_RCVALIGN* and *DD_TXALIGN* in header file *M5475E.H* to be 4.

To avoid data caching issues, the driver also ensures that its receive and transmit buffers are cache line aligned. The driver must always copy the entire received Ethernet frame from its private receive buffer to a KwikNet packet buffer to ensure proper long alignment of the data for processing by KwikNet. The driver must also copy the entire outgoing Ethernet frame from a KwikNet packet buffer to a private transmit buffer to ensure proper alignment of the frame for processing by the MCDMA microcode.

Warning

KwikNet packet buffers must be aligned to meet or exceed this driver's receive alignment requirement. Hence, you must ensure that the "Receive buffer alignment" specified on the Ethernet property page of your KwikNet Network Parameter File is a multiple of the value assigned to *DD_RCVALIGN* in header file *M5475E.H*.

Porting the MCF5475 FEC Ethernet Device Driver

The KwikNet MCF5475 FEC Ethernet Device Driver is ready for use on the Freescale MCF5475 Evaluation Board. Board driver module *KN_B5475.C* is provided with KwikNet for use with this board. With a simple edit to header file *KN5475MD.H* it can also be used on the Freescale MCF5485 Evaluation Board.

To port this device driver to your MCF5475 hardware, proceed as follows. Review header files *M5475E.H* and *KN5475MD.H* to be sure that the symbols defined at the beginning of the files match your intended MCF5475 use. Alterations will be required in only the rarest of configurations.

If necessary, adapt the compiler specific functions which manage the MCF5475 data cache. The code for these functions is defined near the end of header file *M5475E.H*. Driver module *M5475E.C* includes this header file once to access these low level function prototypes. The header file is then included again, near the end of file *M5475E.C*, to generate the code sequences specified at the end of the header file.

If you are using a ColdFire compiler that is not supported by KADAK, you will have to edit the code fragments in header file *M5475E.H* to use the inline assembly language features of your compiler. Look for comments of the form "*Vendor* =" and edit the vendor definitions which most closely match the characteristics of your compiler. Make sure that your definitions are compiled unconditionally and that the definitions for all other vendors are excluded.

If your compiler does not support inline assembly language statements, you will have to provide low level cache manipulation functions in a separate assembly language module.

If you are using AMX CFire, you must use the AMX vector numbers as IRQ identifiers for the interrupting devices FEC0 and FEC1 (103 and 102 respectively).

Configuring the MCF5475 FEC Ethernet Device Driver

Use the KwikNet Configuration Manager to edit your Network Parameter File to include this Ethernet device driver. Assume that you require the following configuration for use on the Freescale MCF5475 Evaluation Board.

MCF5475 FEC0 is to be used operating at 10Mbps.
MCF5475 FEC0 is to be used to access the external MII PHY controller.
Caching must be accommodated.
The Ethernet link must operate in half duplex mode.
PHY device wiring mode is 18-wire (MII).
The controller is to be assigned Ethernet address *0x00CF5475EB00*.
KwikNet IRQ identifier 103 (*0x67*) identifies FEC0 as the interrupt source.
FEC0 interrupt level is 5 and the priority within that level is 0.
PHY id 0 identifies the Ethernet connector with which FEC0 is to be used.
DMA channel 0 is assigned for the MCDMA FEC receive task.
MCDMA FEC receive task priority is 5.
MCDMA FEC receive task initiator is 16 (*0x10*).
DMA channel 1 is assigned for the MCDMA FEC transmit task.
MCDMA FEC transmit task priority is 6.
MCDMA FEC transmit task initiator is 17 (*0x11*).
4 transmit buffers and 8 receive buffers are required.

Enter the following five device driver parameters:

Parameter #1: *0x00CF0000*
Parameter #2: *0x5475EB00*
Parameter #3: *0x00006750*
Parameter #4: *0x11611050*
Parameter #5: *0x00040008*

Generate your KwikNet Library Make File and use it to build your KwikNet Library.

Compile the KwikNet board driver module *KN_B5475.C*.

Create the KwikNet MCF5475 MCDMA Library *KN5475E.A* from the compiled object modules for the device driver module *M5475E.C*, the pin assignment module *M5475EBS.C*, the MCDMA support module *KN5475MD.C* and the MCDMA API and microcode source files. Follow the instructions provided in Chapter 5 of the KwikNet MCF5475 FEC Device Driver User's Guide.

Revise your KwikNet application link specification to include the board driver object module *KN_B5475.O* along with your application modules. Link library module *KN5475E.A* after the KwikNet CFire Library *KN513IP.A* or the KwikNet Porting Kit Library *KN713IP.A*.

The following example illustrates a configuration which uses FEC1 on the Freescale MCF5475 Evaluation Board.

MCF5475 FEC1 is to be used operating at 100Mbps.
MCF5475 FEC0 must be used to access the external MII PHY controller.
Caching must be accommodated.
The Ethernet link must operate in full duplex mode.
PHY device wiring mode is 18-wire (MII).
The controller is to be assigned Ethernet address *0x00CF5475EB01*.
KwikNet IRQ identifier 102 (*0x66*) identifies FEC1 as the interrupt source.
FEC1 interrupt level is 5 and the priority within that level is 1.
PHY id 1 identifies the Ethernet connector with which FEC1 is to be used.
DMA channel 2 is assigned for the MCDMA FEC receive task.
MCDMA FEC receive task priority is 5.
MCDMA FEC receive task initiator is 20 (*0x14*).
DMA channel 3 is assigned for the MCDMA FEC transmit task.
MCDMA FEC transmit task priority is 6.
MCDMA FEC transmit task initiator is 21 (*0x15*).
2 transmit buffers and 4 receive buffers are required.

Enter the following five device driver parameters:

Parameter #1: *0x00CF0109*
Parameter #2: *0x5475EB01*
Parameter #3: *0x00016651*
Parameter #4: *0x15631452*
Parameter #5: *0x00020004*

This page left blank intentionally.