KwikNet®

MPC5200 FEC Device Driver

User's Guide

Version 3

First Printing: November 15, 2004
Last Printing: September 15, 2005

Manual Order Number: PN383-9E

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 MPC5200 FEC Device Driver User's Guide

Table of Contents

This page left blank intentionally.

KwikNet MPC5200 FEC Device Driver

1. Introduction

The Motorola PowerPC MPC5200 includes a single channel Fast Ethernet Controller (FEC) that supports 10Mbps and 100Mbps data transfer rates. The FEC can operate only in conjunction with the MPC5200 BestComm DMA Engine that provides DMA services to a number of MPC5200 subsystems.

The BestComm engine is driven by its private tasks which execute from microcode located in the MPC5200 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 a BestComm task unless otherwise specified.

The BestComm 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 BestComm DMA Engine.

MPC5200 User Manual
BestComm API User's Guide
Application Notes 2251, 2604, 2609

Getting Started

The KwikNet MPC5200 FEC Ethernet Device Driver consists of a number of components which collectively support a single Ethernet network interface 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 MPC5200 BestComm Library. The device driver can be used with KwikNet PPC32 (PN383-2) and AMX PPC32 (PN382-1) or with the KwikNet Porting Kit (PN713-2).

To add the MPC5200 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 MPC5200 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 MPC5200 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 MPC5200 FEC device driver components must be compiled and the resulting object modules must be merged into the KwikNet MPC5200 BestComm Library. Your application is then compiled and linked with this library and the KwikNet Library as described in Chapter 5.

2. Installation

The KwikNet MPC5200 FEC Ethernet Device Driver is provided on the KwikNet CD-ROM. The installation process installs the driver files in directory *KNT303\KN5200E* within an installation directory of your choice. Note that this directory is separate from the KwikNet PPC32 installation directory *KNT383* or the KwikNet Porting Kit installation directory *KNT713*.

The driver files are installed in the following subdirectories within installation directory *KNT303\KN5200E*:

ERR	Construction error summary
M5200E	Driver source and header files
M5200BC	BestComm 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 MPC5200 FEC Ethernet Device Driver consists of the following files:

M5200E\M5200E.H	Device driver header file
M5200E\M5200E.C	Device driver source file
M5200E\M5200EBS.C	Physical device pin assignment and MII setup
M5200E\KN5200BC.C	KwikNet BestComm initialization source file
M5200E\KN5200BC.H	KwikNet BestComm header file
MAKE\KN5200E.MAK	KwikNet MPC5200 BestComm Library make specification file
README.TXT	Version information

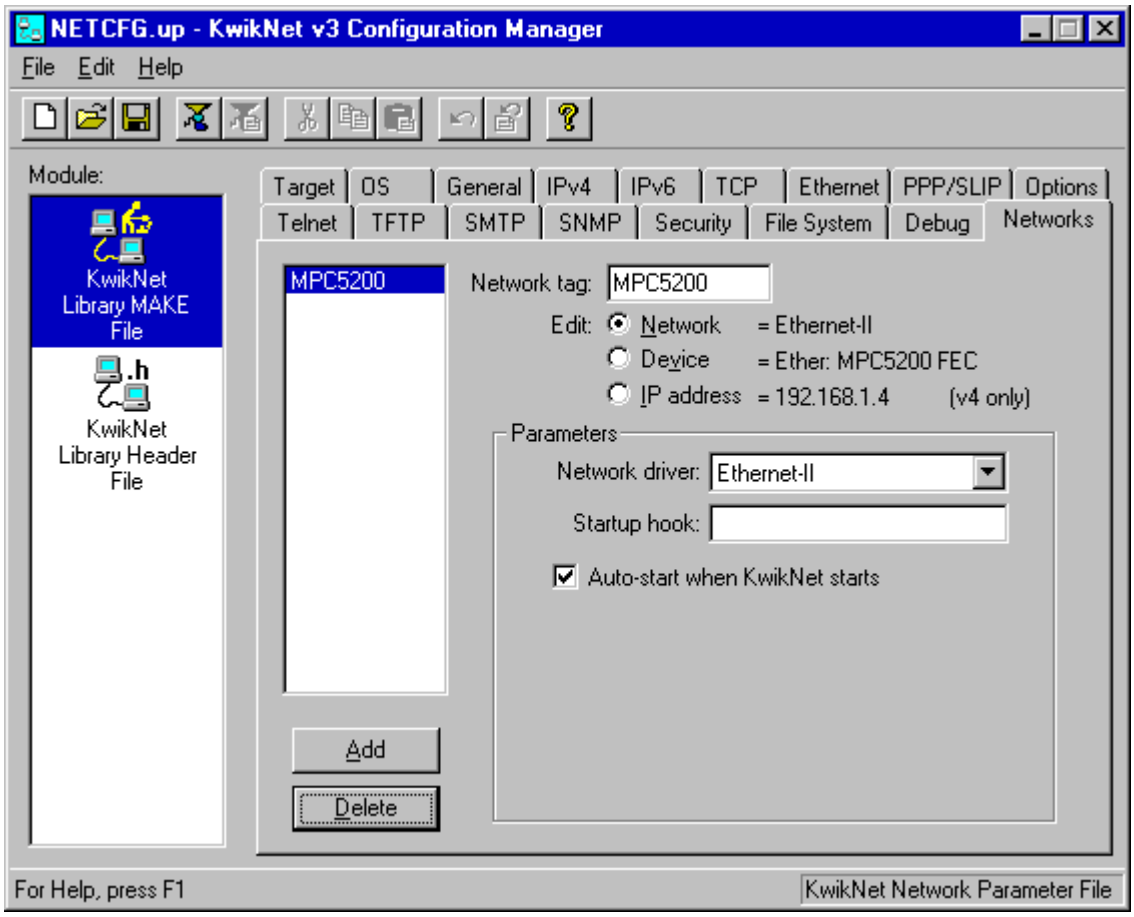
The following BestComm API and microcode source files are available from Freescale Semiconductor, Inc. These files are provided to you as permitted by the terms of the Freescale source code distribution agreement described in the source files. The files provided by KADAK in directory *M5200BC* are the most recent variants tested by KADAK. Text file *README.TXT* in the root of the installation directory identifies the BestComm version and its release date.

M5200BC\BESTCOMM_API.H
M5200BC\BESTCOMM_PRIV.H
M5200BC\DMA_IMAGE.H
M5200BC\DMA_IMAGE.CAPI.H
M5200BC\PPCTYPES.H
M5200BC\MGT5200\MGT5200.H
M5200BC\MGT5200\SDMA.H
M5200BC\TASK_API\BESTCOMM_API_MEM.H
M5200BC\TASK_API\BESTCOMM_CNTRL.H
M5200BC\TASK_API\TASKSETUP_BDTABLE.H
M5200BC\TASK_API\TASKSETUP_GENERAL.H
M5200BC\BESTCOMM_API.C
M5200BC\DMA_IMAGE.C
M5200BC\DMA_IMAGE.RELOC.C
M5200BC\LOAD_TASK.C
M5200BC\TASKSETUP_BDTABLE.C
M5200BC\TASKSETUP_FEC_RX_BD.C
M5200BC\TASKSETUP_FEC_TX_BD.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 MPC5200 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.

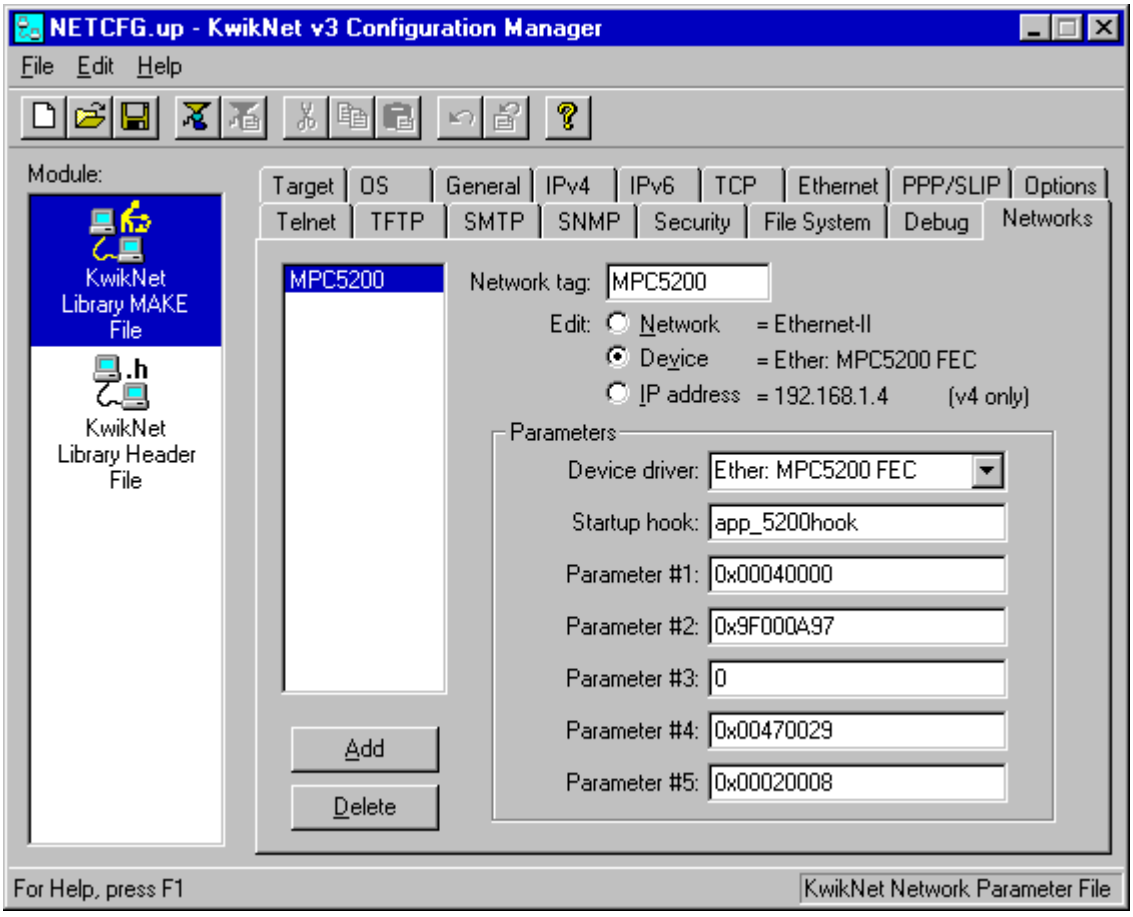


4. Configuring the Device Driver

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 MPC5200 FEC Ethernet Device Driver are defined in Appendix A.



5. Making the KwikNet MPC5200 BestComm Library

The make process depends upon the structure of the KwikNet MPC5200 FEC Ethernet Device Driver installation directory *KN5200E*. When the driver is installed, the following subdirectories are created within directory *KN5200E*.

ERR	Construction error summary
M5200E	Driver source and header files
M5200BC	BestComm 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 MPC5200 BestComm 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 PPC32
TOOLME	*ME*	Metrowerks	KwikNet PPC32
TOOLMW	*MW*	MetaWare	KwikNet PPC32
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 PPC32 . 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 *KN5200E.INC* used to tailor the library construction process specifically for the generation of the KwikNet MPC5200 BestComm 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 PPC32, make sure that your tailoring file *KNZZZCC.INC* from KwikNet installation directory *KNT383\TOOLXXX* matches the copy in driver directory *KN5200E\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 *KN5200E\TOOLUU*. Then edit tailoring file *KN5200E\TOOLUU\KN5200E.INC* to use the compilation and librarian commands for your specific software development tools.

Creating the Driver Library

The KwikNet MPC5200 BestComm Library must be constructed from within the driver installation directory *KN5200E\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 *KN5200E\MAKE* directory the current directory.

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

```
NMAKE -fKN5200E.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 *KN5200E\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 -fKN5200E.MAK "TOOLSET=ME" "TRKPATH=C:\TRECK"
```

During the library construction process, all KwikNet device driver source files and BestComm source files will be compiled and the resulting object modules will be placed in directory *KN5200E\TOOLXXX\LIB*. The KwikNet MPC5200 BestComm Library *KN5200E.A* will be created from these object files and placed in directory *KN5200E\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 MPC5200 BestComm Library file *KN5200E.A* must be linked with your application. Revise your KwikNet application link specification to include this library after the KwikNet Library.

> Note
>
> The BestComm files listed in directory *M5200BC* may vary as new versions of the BestComm software are released. If you receive an updated version of the BestComm source code directly from Freescale, you may have to edit the library make specification file *KN5200E.MAK* to revise the list of BestComm source files being compiled and to correct their dependencies.

6. Special Considerations

6.1 BestComm Initialization

The KwikNet MPC5200 FEC Ethernet Device Driver depends upon the BestComm API and microcode supplied by Freescale Semiconductor, Inc.

The BestComm 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 BestComm tasks upon which the subsystem depends.

Initialization of the BestComm firmware presents the system designer with some difficult startup issues. A number of different parts of an application will depend upon the BestComm firmware. Each part must initialize the BestComm firmware properly for use by that part. The KwikNet MPC5200 FEC Ethernet Device Driver is one such part. And yet the BestComm microcode must be loaded and initialized only once.

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

A BestComm initialization procedure *kn_BestCommSetup()* is provided with this device driver in source file *KN5200BC.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_BestCommSetup()* 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_BestCommSetup()* handles the relocatable loading of the BestComm firmware image into system memory. It then initializes the BestComm API, microcode and private Task Table for subsequent use by the KwikNet MPC5200 FEC Ethernet Device Driver.

You are free to edit source file *KN5200BC.C* to adapt the BestComm firmware loading and initialization sequence to your particular needs. Alternatively, you can merge the code from module *KN5200BC.C* into your own BestComm initialization module.

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

BestComm FEC Initialization

The KwikNet MPC5200 FEC device driver must initialize the BestComm FEC receive and transmit tasks to match the driver's configured requirements. Unfortunately, the BestComm FEC initialization must only be performed once. However, the MPC5200 FEC driver can be opened and closed many times as KwikNet operates. Each time the driver is opened, it must initialize the FEC hardware but it must only initialize the BestComm FEC subsystem once.

To meet this requirement, an FEC initialization procedure *kn_BestCommFEC()* is provided in source file *KN5200BC.C*. The device driver calls this procedure to prepare the BestComm FEC subsystem for subsequent use by the driver. The procedure is called every time that the Ethernet network interface is opened for use.

The function receives a pointer to an initialization structure *knx_bcfecsetup* defined in header file *KN5200BC.H*. This structure serves two purposes. On input, it presents the configuration information needed to initialize the BestComm FEC subsystem. On output, it contains the task identifiers for the BestComm FEC RX and TX tasks.

Procedure *kn_BestCommFEC()* maintains a copy of structure *knx_bcfecsetup* as received the first time that the function is called. The BestComm FEC subsystem is initialized and the results are saved in static variable *kn_bcdata*. On subsequent calls to *kn_BestCommFEC()*, the input parameters are compared to the saved values. If the new parameters do not match the original values, the call is rejected with an error status because the BestComm FEC subsystem cannot be reconfigured to meet the revised configuration requested.

A custom timing service is implemented in source file *KN5200BC.C* for the benefit of the MPC5200 FEC device driver. Precise timing is derived from the internal MPC5200 Time Base Register. 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 time base clock frequency as defined by symbol *TBR_FREQ* in header file *KN5200BC.H*.

You are free to edit source file *KN5200BC.C* to adapt the BestComm FEC initialization sequence to your particular needs. Alternatively, you can merge the code from module *KN5200BC.C* into your own BestComm task initialization module, being sure to retain FEC setup procedure *kn_BestCommFEC()* and its functional integrity.

MPC5200 Register Support

Two low level MPC5200 register access functions are provided in source file *KN5200BC.C*. Function *kn_get_mbar()* reads the Memory Base Address Register (*MBAR*) and returns its value to the caller. Function *kn_get_tbr()* reads the low order 32 bits of the 64-bit Time Base Register (TB) and returns its value to the caller.

These functions are implemented using the assembly language features offered by each of the C compilers tested by KADAK. The functions are implemented in a compiler specific manner in header file *KN5200BC.H*. Instances of the functions are included near the end of source file *KN5200BC.C* by defining symbol *KN_CCNEED_MPC5200SPR* and including header file *KN5200BC.H* for a second time.

6.2 BestComm Source Code Modifications

KADAK has successfully used the BestComm 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 BestComm source code is compiled using the Diab (Wind River) or MetaWare tools, numerous warnings and some errors 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.

The errors occur because of the use of the non-ANSI C keyword *inline* in header file *BESTCOMM_API.H*. This keyword is not supported by either Diab or MetaWare compilers.

To avoid the errors when using these tools, KADAK has inserted the following code fragment near the beginning of header file *BESTCOMM_API.H*:

```
/* KADAK Products Ltd.                                          */
/* Patch to avoid compiler errors.                              */

#ifdef CC_INLINE
/* Define alternate keyword for inline function declaration.    */
#define inline CC_INLINE
#endif
```

Symbol *CC_INLINE*, if required, is defined on the command line used to invoke the compiler in tailoring file *KN5200E.INC*.

If the compiler supports an alternate keyword for an in-line function, symbol *CC_INLINE* is defined to be that keyword. For example, the tailoring file for MetaWare tools defines *CC_INLINE* to be *_Inline*, the equivalent MetaWare keyword.

If the compiler does not support in-line functions, symbol *CC_INLINE* must be defined to be an empty string. For example, the tailoring file for Diab tools defines *CC_INLINE* but does not specify a value for the symbol, since it does not support the feature.

If the compiler supports the *inline* keyword, symbol *CC_INLINE* must not be defined.

> Note
>
> No improper operation of the BestComm API and microcode has been observed as a consequence of this code patch. Note that the BestComm software compiled in this fashion has been exercised using the **FEC subsystem only**.

6.3 MPC5200 Interrupt Exceptions

The PowerPC MPC5200 interrupt management subsystem is incredibly complex. All device interrupt sources are funneled through three PowerPC exceptions in the processor Exception Vector Table. Normal interrupt sources are vectored to the external interrupt exception at offset *0x0500*. Critical interrupts, including all interrupt requests generated by the BestComm DMA Engine, are vectored to the critical interrupt exception at offset *0x0A00*. Other internal interrupt sources are vectored to the system management exception at offset *0x1400*.

Assembly language code must be installed in each of these exception vectors to handle the exception, determine the source of the interrupt which initiated the exception and service the interrupt source.

A further complication arises because the critical interrupt exception cannot be masked off. Hence, a critical interrupt exception from a BestComm interrupt request can preempt service of a normal interrupt exception or a system management exception.

If you are not familiar with the PowerPC architecture, its exception handling rules, its interrupt controller operation and PowerPC assembly language programming, writing exception handler code can be a daunting task. In fact, the complexity provides a good reason to use an RTOS such as KADAK's AMX PPC32 kernel which services these exceptions for you.

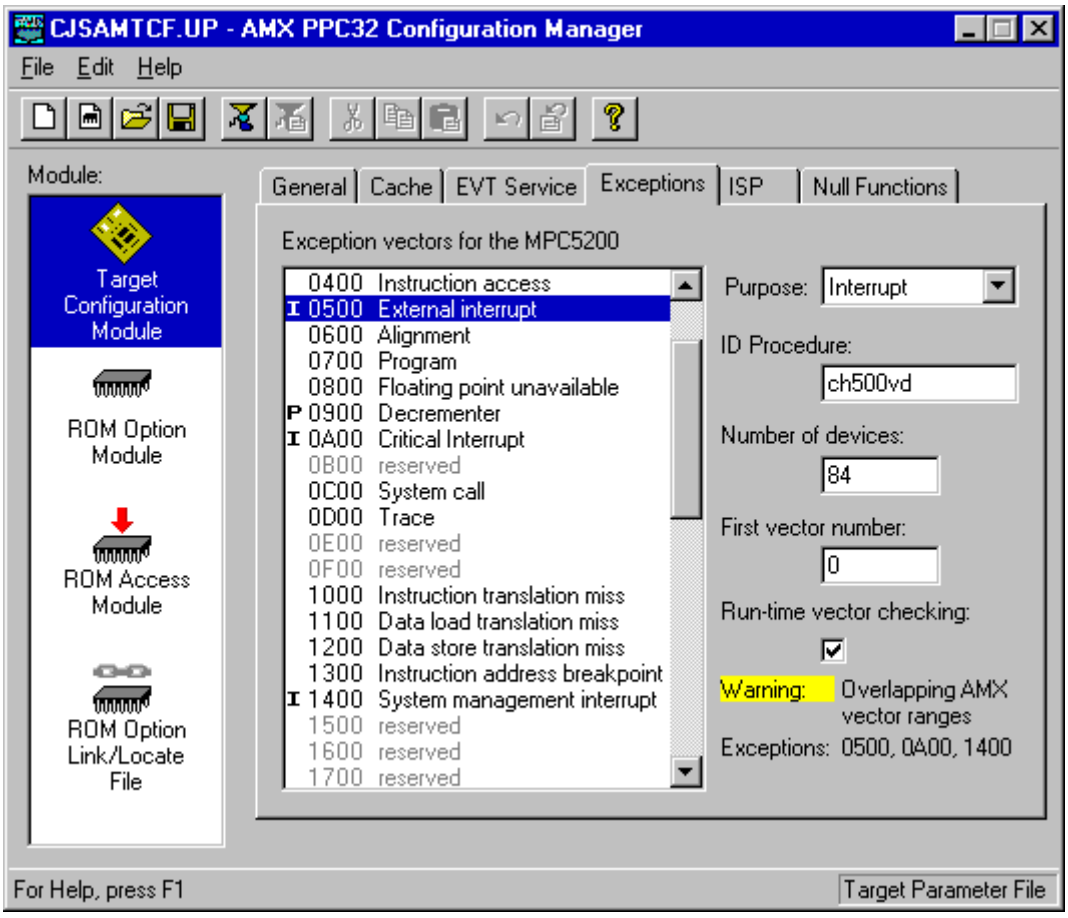
Operation with AMX PPC32

Included with AMX PPC32 is a board support module *LITE5200.S* for the Motorola Lite5200 Evaluation Board. Board initialization function *chbrdinit()* within this module is called prior to launching AMX. This function programs the MPC5200 interrupt controller to reroute the critical interrupt exception from offset *0x0A00* to the external interrupt exception at offset *0x0500*. Doing so resolves the interrupt nesting issue introduced by the non-maskable critical interrupt exception.

Also included in the AMX MPC5200 board support module is the AMX Interrupt Identification Procedure *ch500vd()* which decodes the interrupt source and vectors through the linear AMX Vector Table to the device specific ISP root. The ISP root calls the ISP stem to dismiss the interrupt request and, if necessary, signal that the ISP Handler is to be executed as soon as possible by the AMX Interrupt Supervisor.

Interrupt Identification Procedure *ch500vd()* maps the interrupt sources which generate an external interrupt exception, a critical interrupt exception or a system management exception to a single block of 84 vectors within the AMX Vector Table. Interrupt Identification Procedure *ch500vd()* must be used for each of the three MPC5200 exceptions.

The base vector for the block of interrupt vectors within the AMX Vector Table is specified by you in your AMX Target Configuration Module. Use the AMX Configuration Manager to edit your Target Parameter File and enter your definitions on the Exceptions property page. Note that a Warning will always be displayed indicating that the three MPC5200 exceptions share a common, overlapping region of the AMX Vector Table. The warning can be safely ignored.



The MPC5200 FEC device driver provides support for two interrupt sources: the FEC channel and the BestComm FEC subsystem. Each of these interrupt sources is assigned to a specific vector in the AMX Vector Table.

If the "First vector number" for the external interrupt exception is 0, then the AMX vector number for the FEC interrupt will be 41=0+41.

If the "First vector number" for the critical interrupt exception is 0, then the AMX vector number for the BestComm FEC interrupt will be 71=0+68+3 where 68 is the offset of the 16 AMX vectors reserved for the BestComm DMA Engine and 3 is the BestComm task number of the BestComm Ethernet RX task.

These AMX vector numbers are used as the KwikNet IRQ Identifiers in the device driver parameters specified in the data sheet in Appendix A.

6.4 Configuring the Physical Device

The Ethernet transceiver device (PHY) is configured by function *kn_5200_pcon_fec()* in the driver's pin assignment and board setup module *M5200EBS.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 MPC5200 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 MPC5200 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 MPC5200 FEC device driver's PHY initialization strategy, you must edit the device driver header file *M5200E.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. MPC5200 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 MPC5200 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.

KwikNet®

Device Driver

MPC5200 FEC Ethernet

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

MPC5200 FEC Ethernet
Device Driver Data Sheet

Filename *M5200E.C*, *M5200E.H*, *M5200EBS.C*

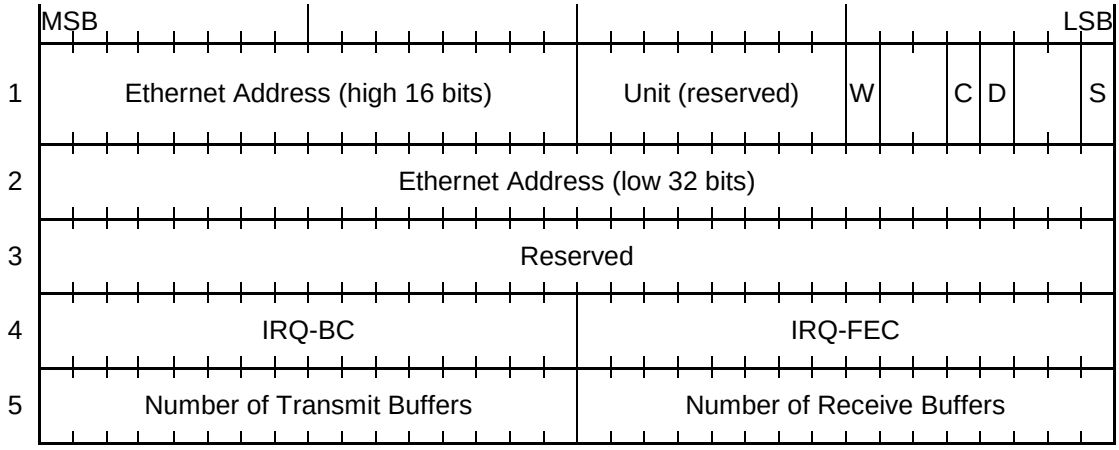
Mnemonic *M5200*

Description The KwikNet MPC5200 FEC Ethernet Device Driver supports the use of the MPC5200 Fast Ethernet Channel (FEC) for 10Mbps or 100Mbps Ethernet communication. The driver depends upon the BestComm DMA Engine for transmission and reception of Ethernet frames via the FEC. The driver is provided ready for use on the Motorola Lite5200 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 *M5200E.H*. The default values and editing instructions are included in the file. These parameters control the following features:

- IP (system) bus clock frequency
- 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 channel to be used for network communication. Since the MPC5200 FEC controller has only one channel, a value of 0 must be used.

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 channel will operate in half duplex mode. If this bit is 1, the FEC channel will operate in full duplex mode.

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 *M5200E.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-BC, IRQ-FEC Parameter IRQ-BC is the KwikNet IRQ identifier which defines how the BestComm Ethernet receive task interrupt requests are vectored. Parameter IRQ-FEC is the KwikNet IRQ identifier which defines how the FEC interrupt request is vectored. These parameters are used by the board driver module *KN_BOARD.C* to map logical device interrupts to the underlying target hardware and operating system.

Note that device interrupts to the PowerPC are funneled through one or more PowerPC exception vectors. Multiple devices can generate interrupts through a single PowerPC exception vector.

AMX PPC32 handles the separation of interrupts and their vectoring through a single linear array referred to as the AMX Vector Table. Consequently, when using KwikNet with AMX PPC32, the AMX vector numbers are used as the KwikNet IRQ identifiers.

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.

Buffers These parameters define the number of transmit and receive buffers which the BestComm DMA Engine must be configured to support. Descriptors for these transmit and receive buffers will be allocated in the BestComm Buffer Descriptor Tables. Unless you edit the definitions in header file *M5200E.H*, you must specify a minimum of one transmit buffer and two receive buffers. Two transmit buffers and eight receive buffers are frequently sufficient.

BestComm FEC descriptors can only be allocated once. Hence, the descriptors are allocated the first time that the network interface is opened. For prebuilt networks, the descriptors will be allocated the first time that KwikNet is started. For networks added at runtime, the descriptors will be allocated upon your first call to KwikNet procedure *kn_ifopen()* after having added a network interface with a call to *kn_ifadd()*.

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 BestComm 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 *M5200E.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 BestComm 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 *M5200E.H*.

Porting the MPC5200 FEC Ethernet Device Driver

The KwikNet MPC5200 FEC Ethernet Device Driver is ready for use on the Motorola Lite5200 Evaluation Board. The default board driver module *KN_BOARD.C* provided with KwikNet is also ready for use with this board.

To port this device driver to your MPC5200 hardware, proceed as follows. Review header file *M5200E.H* and be sure that the symbols defined at the beginning of the file match your intended MPC5200 use. Alterations will be required in only the rarest of configurations.

If you are using AMX PPC32, you can use Interrupt Identification Procedure *ch500vd()* in the AMX board support module, *LITE5200.S*, to derive the AMX vector number for the interrupting devices.

Configuring the MPC5200 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 Motorola Lite5200 Evaluation Board.

MPC5200 FEC is to be used operating at 10Mbps.
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 *0x00049F000A97*.
KwikNet IRQ identifier 71 (*0x47*) identifies the BestComm Ethernet RX task as the interrupt source.
KwikNet IRQ identifier 41 (*0x29*) identifies FEC channel 0 as the interrupt source.
2 transmit buffers and 8 receive buffers are required.

Enter the following five device driver parameters:

Parameter #1: *0x00040000*
Parameter #2: *0x9F000A97*
Parameter #3: *0*
Parameter #4: *0x00470029*
Parameter #5: *0x00020008*

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

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

Create the KwikNet MPC5200 BestComm Library *KN5200E.A* from the compiled object modules for the device driver module *M5200E.C*, the pin assignment module *M5200EBS.C*, the BestComm support module *KN5200BC.C* and the BestComm API and microcode source files. Follow the instructions provided in Chapter 5 of the KwikNet MPC5200 FEC Device Driver User's Guide.

Revise your KwikNet application link specification to include the board driver object module *KN_BOARD.O* along with your application modules. Link library module *KN5200E.A* after the KwikNet PPC32 Library *KN383IP.A* or the KwikNet Porting Kit Library *KN713IP.A*.