DYNAMIC ENGINEERING
150 DuBois, Suite C
Santa Cruz, CA 95060
(831) 457-8891
http://www.dyneng.com
sales@dyneng.com
Est. 1988

IpRelay16

Win10 Driver Documentation For the IP-Relay16 IP module

Developed with Windows Driver Foundation Ver1.25

Manual Revision 1.0
Corresponding Firmware: Design ID 1, Revision 1.0
Corresponding Hardware: 10-2019-1401

IpRelay16
WDF Device Driver for the
IP-Relay-16 IP module

Dynamic Engineering
150 DuBois, Suite C
Santa Cruz, CA 95060
(831) 457-8891
FAX: (831) 457-4793

©2020 by Dynamic Engineering.
Other trademarks and registered trademarks are owned by their respective manufacturers.
Revised March 23, 2020

This document contains information of proprietary interest to Dynamic Engineering. It has been supplied in confidence and the recipient, by accepting this material, agrees that the subject matter will not be copied or reproduced, in whole or in part, nor its contents revealed in any manner or to any person except to meet the purpose for which it was delivered.

Dynamic Engineering has made every effort to ensure that this manual is accurate and complete. Still, the company reserves the right to make improvements or changes in the product described in this document at any time and without notice. Furthermore, Dynamic Engineering assumes no liability arising out of the application or use of the device described herein.

The electronic equipment described herein generates, uses, and can radiate radio frequency energy. Operation of this equipment in a residential area is likely to cause radio interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Dynamic Engineering’s products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Dynamic Engineering.

Connection of incompatible hardware is likely to cause serious damage.

Table of Contents

Introduction

The IpRelay16 driver is a Windows device driver for the IP-Relay16 IP module from Dynamic Engineering. This driver was developed with the Windows Driver Foundation version 1.25 (WDF) from Microsoft, specifically the Kernel-Mode Driver Framework (KMDF).

The IP module has a Xilinx XC-95144XL CPLD to implement the IP interface and 16 SPDT (FORM C) relays that are controlled over the IP bus. The IP bus is implemented by an IP carrier board.

When the carrier board is recognized by the PCI[e] bus configuration utility it will load the appropriate IP carrier driver which implements and enumerates an IP bus. The identity of the IP modules installed on the carrier is determined and the appropriate IP module drivers are loaded. The module driver creates a device object for each of the IP-Relay-16 IP modules and initializes the hardware. IO Control calls (IOCTLs) are used to configure the board, control the relays and read their status.

Note

This documentation will provide information about all calls made to the drivers, and how the drivers interact with the device for each of these calls. For more detailed information on the hardware implementation, refer to the IP-Relay16 hardware manual and the respective carrier software and hardware manuals.

Driver Installation

Note: An appropriate IP carrier driver must be installed before any IP modules can be detected by the system.

There are several files provided in each driver installation package. These files include IpRelay16.cat, IpRelay16.sys and IpRelay16.inf. Also IpRelay16Public.h and IpPublic.h are C header files that define the Application Program Interface (API) for the IpRelay16 driver. These file are required at compile time by any application that wishes to interface with the driver, but are not needed for driver installation.

Windows 10 Installation

Copy IpRelay16.inf, IpRelay16.cat, IpRelay16.sys to a removable memory device, or another accessible location if preferred.

With the IP-Relay16 hardware installed, power-on the host computer.

- Open the ***Device Manager*** from the control panel.
- Under ***Other devices*** there should be, for each valid IP module installed, a device icon with an index appended carrier device name followed by an IP Slot designation where the module is installed*.
- Right-click on the target device and select ***Update Driver Software***.
- Insert the removable memory device prepared above if necessary.
- Select ***Browse my computer for driver software***.
- Select ***Browse*** and navigate to the location where the appropriate files are stored.
- Select ***Next***. The appropriate IP device driver should now be installed.
- Select ***Close*** to close the update window.

This process must be completed for each new IP device that is installed.

* If no IP devices are displayed, check to see that an IP Carrier Device is present in the Device Manager and click on the ***Scan for hardware changes*** icon on the tool-bar or select it in the Action menu.

Driver Startup

Once the driver has been installed it will start automatically when the system recognizes the hardware. A handle can be opened to a specific board by using the CreateFile() function call and passing in the device name obtained from the system. The interface to the device is identified using a globally unique identifier (GUID), which is defined in IpRelay16Public.h. See main.c in the example IpRelay16UserApp project for an example of how to acquire a handle for the device.

Note: In order to build an application you must link with setupapi.lib.

IO Controls

The drivers use IO Control calls (IOCTLs) to configure the device. IOCTLs refer to a single Device Object, which controls a single IP module. IOCTLs are called using the Win32 function DeviceIoControl() (see below), and passing in the handle to the device opened with CreateFile() (see above). IOCTLs generally have input parameters, output parameters, or both. Often a custom structure is used.

```
BOOL DeviceIoControl(
  HANDLE        hDevice,          // Handle opened with CreateFile()
  DWORD         dwIoControlCode,  // Control code defined in API header file
  LPVOID        lpInBuffer,       // Pointer to input parameter
  DWORD         nInBufferSize,    // Size of input parameter
  LPVOID        lpOutBuffer,      // Pointer to output parameter
  DWORD         nOutBufferSize,   // Size of output parameter
  LPDWORD       lpBytesReturned,  // Pointer to return length parameter
  LPOVERLAPPED  lpOverlapped,     // Optional pointer to overlapped structure
);                                //   used for asynchronous I/O
```

The IOCTLs defined for the IpRelay16 driver are described below:

IOCTL_IP_RELAY_16_GET_INFO

Function: Returns the current driver revision, instance number, module location and other carrier information.
Input: None
Output: DRIVER_IP_DEVICE_INFO structure
Notes: This call does not access the hardware, only driver parameters. CarrierSwitch returns the value of the 8-position IP carrier dip-switch selection when this IP was enumerated. See the definition of DRIVER_IP_DEVICE_INFO below.

```
 // Driver, design revision, instance/slot and other information
typedef struct _DRIVER_IP_DEVICE_INFO {
   UCHAR    DriverRev;         // Driver revision
   UCHAR    FirmwareRev;       // Firmware major revision
   UCHAR    FirmwareRevMin;    // Firmware minor revision
   UCHAR    InstanceNum;       // Zero-based device number
   UCHAR    CarrierSwitch;     // 0..0xFF
   UCHAR    CarrierSlotNum;    // 0..7 -> IP slots A, B, C, D, E, F, G or H
   UCHAR    CarDriverRev;      // Carrier driver revision
   UCHAR    CarFirmwareRev;    // Carrier firmware major revision
   UCHAR    CarFirmwareRevMin;// Carrier firmware minor revision
   UCHAR    CarCPLDRev;        //**Used for PCIe carriers only** 0xFF for others
   UCHAR    CarCPLDRevMin;     //**Used for PCIe carriers only** 0xFF for others
   BOOLEAN  Ip32MCapable;      // IP is capable of both 8MHz and 32MHz operation
   BOOLEAN  NewIpCntl;         // New IP slot control interface
   WCHAR    LocationString[IP_LOC_STRING_SIZE];
} DRIVER_IP_DEVICE_INFO, *PDRIVER_IP_DEVICE_INFO;

#define IP_LOC_STRING_SIZE 25 // Maximum size of location string (WCHARs)
```

IOCTL_IP_RELAY_16_SET_IP_CONTROL

Function: Sets the control configuration of the module's IP slot.
Input: IP_SLOT_CONTROL structure
Output: None
Notes: Specifies the IP clock speed, data access and other control parameters for the IP slot that the board occupies. See the definition of IP_SLOT_CONTROL below.

```
typedef struct _IP_SLOT_CONTROL {
   BOOLEAN  Clock32Sel;
   BOOLEAN  ClockDis;
   BOOLEAN  ByteSwap;
   BOOLEAN  WordSwap;
   BOOLEAN  WrIncDis;
   BOOLEAN  RdIncDis;
   UCHAR    WrWordSel;
   UCHAR    RdWordSel;
   BOOLEAN  BsErrTmOutSel;
   BOOLEAN  ActCountEn;
} IP_SLOT_CONTROL, *PIP_SLOT_CONTROL;
```

IOCTL_IP_RELAY_16_GET_IP_STATE

Function: Returns the control configuration of the module's IP slot plus interrupt and bus error status.
Input: None
Output: IP_SLOT_STATE structure
Notes: Returns the slot control configuration from the previous call along with interrupt enable and activity information. See the definition of IP_SLOT_STATE below.

```
typedef struct _IP_SLOT_STATE {
   BOOLEAN  Clock32Sel;
   BOOLEAN  ClockDis;
   BOOLEAN  ByteSwap;
   BOOLEAN  WordSwap;
   BOOLEAN  WrIncDis;
   BOOLEAN  RdIncDis;
   UCHAR    WrWordSel;
   UCHAR    RdWordSel;
   BOOLEAN  BsErrTmOutSel;
   BOOLEAN  ActCountEn;
 // Slot Status
   BOOLEAN  IpInt0En;
   BOOLEAN  IpInt1En;
   BOOLEAN  IpBusErrIntEn;
   BOOLEAN  IpInt0Actv;
   BOOLEAN  IpInt1Actv;
   BOOLEAN  IpBusError;
   BOOLEAN  IpForceInt;
   BOOLEAN  WrBusError;
   BOOLEAN  RdBusError;
} IP_SLOT_STATE, *PIP_SLOT_STATE;
```

IOCTL_IP_RELAY_16_GET_REVISION

Function: Returns the major and minor revisions of the IP-Relay16 firmware.
Input: None
Output: IP_RELAY_16_REVISION structure
Notes: Reads and returns the firmware revision.

```
// IpRelay16 revision structure
typedef struct _IP_RELAY_16_REVISION {
   USHORT   MajorRev;   // IpRelay16 firmware major revision
   USHORT   MinorRev;   // IpRelay16 firmware minor revision
} IP_RELAY_16_REVISION, *PIP_RELAY_16_REVISION;
```

IOCTL_IP_RELAY_16_SET_RELAY_CONFIG

Function: Sets the relay control configuration for the IP-Relay16.
Input: Relay configuration value (unsigned short integer)
Output: None
Notes: The 16-bit input parameter specifies the configuration for the modules 16 relays. A one in bit position n directs relay n to be set. A zero directs the corresponding relay to be not set.

IOCTL_IP_RELAY_16_GET_RELAY_CONFIG

Function: Returns the relay control configuration for the IP-Relay16.
Input: None
Output: Relay configuration value (unsigned short integer)
Notes: The 16-bit output parameter reports the configuration for the modules 16 relays. A one in bit position n indicates that relay n is set. A zero indicates that the corresponding relay is not set.

IOCTL_IP_RELAY_16_CLOSE_SINGLE_RELAY

Function: Causes a single relay out of 16 to be set.
Input: Relay number (0..15) (unsigned character)
Output: None
Notes: Causes the relay represented by the input parameter to be set (turned on).

IOCTL_IP_RELAY_16_OPEN_SINGLE_RELAY

Function: Causes a single relay out of 16 to be not set.
Input: Relay number (0..15) (unsigned character)
Output: None
Notes: Causes the relay represented by the input parameter to be not set (turned off).

IpRelay16UserApp

This is a simple test application to demonstrate the functionality of the relay module using a test fixture with 16 green LEDs and 16 red LEDs. A green LED is illuminated when the corresponding relay is in the off position and a red LED is illuminated when the corresponding relay is in the on position.

Test 1 exercises all the relays by starting with all relays off. Then all relays are switched on. After a two second delay, the even numbered relays are switched off leaving the odd numbered relays on. After another two second delay, the odd numbered relays are switched off and the even numbered relays are switched on. After another two second delay, all relays are switched off. The test then proceeds to close single relays starting with relay zero until relay 15. There is a half-second delay between steps. This is repeated, turning off relay zero through relay 15. Then repeating single relay (0-15) on and finally single relay (0-15) off. After each step of this process, the configuration of the relays is read and compared to the intended value. If this value does not match at any point, an error is returned.

Test 2 operates on a single relay at a time. The test asks for a relay number between 0 and 15. The corresponding relay is active (turned on). When the space bar is hit the relay will be inactive (turned off). If a ‘q’ is entered the test completes, but if enter is hit with no input the same relay will be active again. This process continues until ‘q’ is entered. This test is provided to check the function of a single relay or whatever is connected to it.

When the test starts the value returned from the get revision call is printed at the top of the menu display.

Warranty and Repair

Please refer to the warranty page on our website for the current warranty offered and options.
http://www.dyneng.com/warranty.html

Service Policy

Before returning a product for repair, verify as well as possible that the driver is at fault. The driver has gone through extensive testing, and in most cases it will be "cockpit error" rather than an error with the driver. When you are sure or at least willing to pay to have someone help then call or e-mail and arrange to work with an engineer. We will work with you to determine the cause of the issue.

Support

The software described in this manual is provided at no cost to clients who have purchased the corresponding hardware. Minimal support is included along with the documentation. For help with integration into your project please contact sales@dyneng.com for a support contract. Several options are available. With a contract in place Dynamic Engineers can help with system debugging, special software development, or whatever you need to get going.

For Service Contact:

Customer Service Department
Dynamic Engineering
150 DuBois, Suite C Santa Cruz, CA 95060
(831) 457-8891
support@dyneng.com

All information provided is Copyright Dynamic Engineering.