NetBurner

Networking in 1 Day!

NBWIFIIN-100CR

Wireless Network Interface Controller

Programmers Guide

Table of Contents

1. Introduction

The NBWIFIIN-100CR is a wireless network interface controller (NIC) with the following features:

- IEEE 802.11 b/g/n.
- Secure data communications with 128-bit WEP, WPA-PSK (TKIP), WPA2-PSK Authentication DSSS with DBPSK and DQPSK, CCK modulations and demodulations supported with long and short preamble.
- Communicates with NetBurner modules via the QSPI interface.
- Can connect to an access point in infrastructure mode.
- Can operate as an access point and also act as a DHCP server for connecting clients.

Please refer to the Getting Started Guide and data sheet on the NBWIFIN product page at www.netburner.com for hardware specific details.

2. Interface Signals

The default hardware and software configuration uses the following NetBurner processor module signals:

- QSPI MOSI, QSPI MISO, QSPI CLK, QSPI CS1
- IRQ 3

Please refer to the Getting Started Guide for additional details and alternate configurations.

3. Additional Documentation

This document covers the wireless product only. If you need assistance in using the NetBurner development tools, TCP/IP stack or Real-Time Operating system, please refer to the following documents located in c:\nburn\docs:

- The Wifi Getting Started Guide and NBWIFIIN data sheet, located on the Wireless products page at www.netburner.com. It contains hardware connector and configuration information.
- NNDK Programmer's Guide: a textbook style document on programming NetBurner devices.
- NBEclipse Getting Started Guide: a tutorial on using the NBEclipse Integrated Development Environment.
- NetBurner Runtime Libraries: API function reference documents for the TCP/IP Stack and RTOS.
- The primary header files for the Wifi functions are nbWifi.h and nbWifiDriver.h, located in the \nburn\include\nbwifi.directory.

4. Implementation Notes

You must call the InitializeStack() routine before calling any Wifi functions.

The standard I/O listen() function listens for data on all active interfaces simultaneously. Replies are sent the same interface from which they were received. So a web page request received on the Wifi interface, will be replied to on the Wifi Interface.

The default network interface is the Ethernet interface. To make an outgoing connection to the Wifi interface use the connectvia() instead of the connect() function.

5. The Wifi Interface Object

The Wifi interface is an object. NBWifi is the Wifi driver namespace, and Master is the name of the class within the namespace. To interact with the Wifi object you need to obtain a pointer to the Wifi object created by the system. Interaction with the Wifi object is done through the object's member functions. If you are not familiar with C++ do not worry, the interface is very straight forward. Once you have the pointer to the object, you call functions similar as you would in C.

```
NBWifi::Master *pNBWifiObject = nullptr;
```

The name of the pointer in this example is pNBWifiObject. You can change the name to anything you like. To set the value of the pointer the Wifi interface must be initialized. For example:

```
// Initialize and get the interface number
int ifnumWifi = WifiInitScanAndShow_SPI();

if (ifnumWifi > 0)
{
    pNBWifiObject = NBWifi::Master::GetDriverByInterfaceNumber( ifnumWifi )
}
else
{
    iprintf("Failed to initialize wifi interface\r\n");
}
```

Once you have the Wifi object pointer you call the member functions with the pointer to the Wifi object prefix as shown below for the Connected() member function.

```
bool connected = pNBWifiObject->Connected();

if ( connected )
{
    Iprintf("Wifi is connected\r\n");
}
```

6. Initialization Functions

The initialization functions below are normally called without parameters so they use the default values for the specific platform. The default values are located in \nburn\<platform>\include\nbwifi\nbWifiDefs.h. If you change any of the signals to suit your custom design you will need to specify the parameters in the function call.

Parameters:

irqNum	Interrupt number used by the microprocessor.
moduleNum	Specifies QSPI module for platforms with multiple QSPI peripherals.
csNum	QSPI chip select number.
connectorNum	Physical connector of the module on which the signal pins are located.
gpioPinNum	Specifies a GPIO pin to be used as a QSPI chip select. Used only for platforms that do not have a native QSPI chip select available.
resetPinNum	Specifies pin to be connect to the reset input of the Wifi module.

Examples:

```
int ifNum = -1;

// Initialize with default parameters
ifNum = WifiInitScanAndShow_SPI();

// Example of specified initialization on MOD54415
// IRQ = 3, QSPI module number 1, QSPI chip select 0,
// Module header J2, GPIO pin not used, reset pin number
// J2.42.
ifNum = WifiInitScanAndShow_SPI(3, 1, 0, 2, -1, 42);

// Initialize with specified SSID/password and
// default IRQ/pin values
ifNum = InitWifi_SPI(“mySSID”, “myPassword”);
```

6.1 WifiInitScan_SPI

Header File:
#include <nbwifi\nbWifi.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
nbWifiScanResult * WifiInitScan_SPI(
                int irqNum          = -1,
                int moduleNum       = -1,
                int csNum           = -1,
                int connectorNum    = -1,
                int gpioPinNum      = -1,
                int resetPinNum     = -1
                );
```

Description:
Initializes Wifi on the SPI bus and performs a scan for access points. The result of the scan is returned as a linked list that contains the scan results. If the WiFi driver has already been initialized by a previous call to one of the WifiInit variations, then only a scan will be performed.

Parameters:

irqNum	Interrupt number used by the microprocessor.
moduleNum	Specifies QSPI module for platforms with multiple QSPI peripherals.
csNum	QSPI chip select number.
connectorNum	Physical connector of the module on which the signal pins are located.
gpioPinNum	Specifies a GPIO pin to be used as a QSPI chip select. Used only for platforms that do not have a native QSPI chip select available.
resetPinNum	Specifies pin to be connect to the reset input of the Wifi module.

Returns:
Returns a pointer to the head of a linked list of scan results.

Notes:
```
struct _STRUCT_PACKED nbWifiScanResult {
    uint8_t lastAndBand;
    uint8_t bssType;
    uint8_t channel;
    uint8_t security;
    uint8_t cipher;
    uint8_t ssidLength;
    int16_t rssi;
    MACADR bssid;
    char ssid[SSID_MAX_LEN + 1];
    struct nbWifiScanResult * next;
};
```

`nbWifiScanResult` is defined in nburn/include/nettypes.h.

6.2 WifiInitScanAndShow_SPI

Header File:
#include <nbwifi\nbWifi.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
int WifiInitScanAndShow_SPI(
                int irqNum          = -1,
                int moduleNum       = -1,
                int csNum           = -1,
                int connectorNum    = -1,
                int gpioPinNum      = -1,
                int resetPinNum     = -1
                );
```

Description:
Initialize WiFi on the SPI interface, scan, and display scan results on console serial port.

Parameters:

irqNum	Interrupt number used by the microprocessor.
moduleNum	Specifies QSPI module for platforms with multiple QSPI peripherals.
csNum	QSPI chip select number.
connectorNum	Physical connector of the module on which the signal pins are located.
gpioPinNum	Specifies a GPIO pin to be used as a QSPI chip select. Used only for platforms that do not have a native QSPI chip select available.
resetPinNum	Specifies pin to be connect to the reset input of the Wifi module.

Returns:
Returns the WiFi interface number, if successful. Otherwise, returns an error code < 0. Error codes are defined in nburn/include/nbwifi/nbWifiConstants.h

6.3 InitWifi_SPI

Header File:
#include <nbwifi\nbWifi.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
int InitWifi_SPI(
                const char * SSID        = "",
                const char * password    = "",
                int irqNum               = -1,
                int moduleNum            = -1,
                int csNum                = -1,
                int connectorNum         = -1,
                int gpioPinNum           = -1,
                int resetPinNum          = -1
                );
```

Description:
Initialize WiFi on the SPI bus and attempt to connect to an access point. If the SSID and password are null the values specified in the system configuration will be used. You can call this function with only the SSID and password:

Parameters:

SSID	Network name to connect to.
password	Pre-shared key for the network.
irqNum	Interrupt number used by the microprocessor.
moduleNum	Specifies QSPI module for platforms with multiple QSPI peripherals.
csNum	QSPI chip select number.
connectorNum	Physical connector of the module on which the signal pins are located.
gpioPinNum	Specifies a GPIO pin to be used as a QSPI chip select. Used only for platforms that do not have a native QSPI chip select available.
resetPinNum	Specifies pin to be connect to the reset input of the Wifi module.

Returns:
Returns the WiFi interface number, if successful. Otherwise, returns an error code < 0. Error codes are defined in nburn/include/nbwifi/nbWifiConstants.h

6.4 InitAP_SPI

Header File:
#include <nbwifi\nbWifi.h>		// Found in C:\nburn\include\nbwifi

Synopsis:
```
int InitAP_SPI(
                const char * SSID        = "",
                const char * password    = "",
                uint8_t      channel     = NBWIFI_DEFAULT_WIFICHANNEL,
                int irqNum               = -1,
                int moduleNum            = -1,
                int csNum                = -1,
                int connectorNum         = -1,
                int gpioPinNum           = -1,
                int resetPinNum          = -1
                );
```

Description:
Initialize WiFi on the SPI bus and in access point mode. If the SSID and password are null the values specified in the system configuration will be used. You can call this function with only the SSID and password specified.

Parameters:

SSID	Network name to connect to.
password	Pre-shared key for the network.
Channel	802.11 channel to create the WiFi access point on, 1-11.
irqNum	Interrupt number used by the microprocessor.
moduleNum	Specifies QSPI module for platforms with multiple QSPI peripherals.
csNum	QSPI chip select number.
connectorNum	Physical connector of the module on which the signal pins are located.
gpioPinNum	Specifies a GPIO pin to be used as a QSPI chip select. Used only for platforms that do not have a native QSPI chip select available.
resetPinNum	Specifies pin to be connect to the reset input of the Wifi module.

Returns:
Returns the WiFi interface number, if successful. Otherwise, returns an error code < 0. Error codes are defined in nburn/include/nbwifi/nbWifiConstants.h

7. Connection Functions

7.1 ConnectToAP

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:

```
int ConnectToAP( const char *ssid       = NULL,
                 const char *passwd     = "",
                 uint8_t    retryCount = CONNECT_RETRIES,
                 uint8_t    ssidLen    = 0 );
```

Description:
Connect to an access point. If the SSID and Pass Phrase are not specified, values from the system configuration record will be used.

Parameters:

Ssid	Service Set Identity (SSID); If NULL, uses the SSID stored in the configuration record.
Passwd	Pre-shared key to obtain access to the network; If NULL, uses the password stored in the configuration record.
retryCount	Amount of attempts to try to connect.
ssidLen	Length of the SSID string not including NULL terminating characters.

Returns:
Returns the WiFi interface number, if successful. Otherwise, returns an error code < 0. Error codes are defined in nburn/include/nbwifi/nbWifiConstants.h

7.2 Disconnect

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
void Disconnect();
```

Description:
Disconnect from the network and disable the access point.

Parameters:
None.

Returns:
Nothing.

8. Scan Functions

8.1 Scan

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
nbWifiScanResult * Scan( const char *ssid        = NULL,
                         uint8_t    optionCount = 0,
                         uint16_t   *optionList = NULL);
```

Description:
Execute a synchronous scan, which will block until the scan is complete. A SSID value of NULL means scan for all networks. These functions are not reentrant and results are only valid until the next call to Scan() .

Parameters:

ssid	Service Set Identity (SSID); If NULL, scan for all networks. Otherwise, scan for this network only.
optionCount	Count of options used to filter scan results. This is an optional parameter unless optionList is used.
optionList	List of options used to filter scan results. This is an optional parameter. Passing NULL will simply show all scan results. Option list values can be found in nburn/include/nbwifi/nbWifiConstants.h.

Returns:
Returns a pointer to a linked list of nbWifiScanResult structures.

Notes:
```
struct _STRUCT_PACKED nbWifiScanResult {
    uint8_t lastAndBand;
    uint8_t bssType;
    uint8_t channel;
    uint8_t security;
    uint8_t cipher;
    uint8_t ssidLength;
    int16_t rssi;
    MACADR bssid;
    char ssid[SSID_MAX_LEN + 1];
    struct nbWifiScanResult * next;
};
```

`nbWifiScanResult` is defined in nburn/include/nettypes.h.is defined in nburn/include/nbwifi/nbWifiDriver.h.

8.2 StartAsyncScan

Header File:
#include <nbwifi\nbWifiDriver.h> // Found in C:\nburn\include\nbwifi

Synopsis:
```
int StartAsyncScan( ReceiveScanResultFunc pfCallbackFunc,
                    const char            *ssid       = NULL,
                    uint8_t               optionCount = 0,
                    uint16_t              *optionList = NULL );
```

Description:
An asynchronous scan will operate in the background and execute the specified callback function on completion. If the underlying WiFi driver has an internally timed scan then the callback function will receive an AP with SSID = NULL as the last indicator. Otherwise you should call StopAsyncScan().

Parameters:

pfCallbackFunc	Callback function called when the asynchronous scan is complete.
ssid	Service Set Identity (SSID); If NULL, scan for all networks. Otherwise, scan for this network only.
optionCount	Count of options used to filter scan results. This is an optional parameter unless optionList is used.
optionList	List of options used to filter scan results. This is an optional parameter. Passing NULL will simply show all scan results. Option list values can be found in nburn/include/nbwifi/nbWifiConstants.h.

Returns:
Returns a value of 1 on success.

8.3 StopAsyncScan

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
Stop an asynchronous scan.

Description:
```
void StopAsyncScan();
```

Parameters:
Nothing.

Returns:
Nothing.

9. Utility Functions

9.1 GetSystemInterfaceNumber

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
int GetSystemInterfaceNumber() const;
```

Description:
Get the system interface number of the WiFi interface.

Parameters:
Nothing.

Returns:
Returns the interface number of the WiFi interface.

9.2 StoreSSIDPWToConfig

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
int StoreSSIDPWToConfig( char *ssid, char *password, int ssidLen = 0);
```

Description:
Store an SSID and password into the configuration record, which is in non-volatile memory.

Parameters:

ssid	Service Set Identity (SSID) to store in the config record. Max length of SSID_MAX_LEN, defined in nbwifi/include/nbwifi/nbWificonstants.h.
password	Pre-shared key. Password must be between 8 and 64 characters, or NULL for an open network.
ssidLen	Length of the ssid string not including NULL terminating characters.

Returns:
Returns 1, if successful. Otherwise, returns <= 0. Error codes can be found in nburn/include/nbwifi/nbWifiConstants.h

9.3 GetSSIDFromConfig

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
int GetSSIDFromConfig( char *returnBuf, int maxLen );
```

Description:
Get the Service Set Identity (SSID) stored in the configuration record, which is in non-volatile memory.

Parameters:

returnBuf	A pointer to the buffer used by this function to store the SSID.
maxLen	The size limit for writing to the returnBuf paramter. This is usually the size of returnBuf in bytes.

Returns:
Returns the length of the copied string. Otherwise, returns < 0. Error codes can be found in nburn/include/nbwifi/nbWifiConstants.h.

9.4 GetKeyFromConfig

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
int GetKeyFromConfig( char *returnBuf, int maxLen );
```

Description:
Get the password stored in the configuration record, which is in non-volatile memory.

Parameters:

returnBuf	A pointer to the buffer used by this function to store the password.
maxLen	The size limit for writing to the returnBuf paramter. This is usually the size of returnBuf in bytes.

Returns:
Returns the length of the copied string. Otherwise, returns < 0. Error codes can be found in nburn/include/nbwifi/nbWifiConstants.h.

10. Status Functions

10.1 GetCurSSID

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
int GetCurSSID(char * buf, int maxlen);
```

Description:
Get the Service Set Identity (SSID) of the network you are currently connected to or the SSID of the access point that is currently established.

Parameters:

buf	A pointer to the buffer used by this function to store the SSID.
maxLen	The size limit for writing to the returnBuf paramter. This is usually the size of returnBuf in bytes.

Returns:
Returns the length of the copied string. Otherwise, returns < 0. Error codes can be found in nburn/include/nbwifi/nbWifiConstants.h.

10.2 GetCurBSSID

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
`MACADR GetCurBSSID();`

Description:
Get the basic service set identifier (BSSID) of the connected network.

Parameters:
None.

Returns:
A MACADR object which contains the 48-bit BSSID value, if successful. Otherwise, returns 0.

Notes:
```
typedef struct MACADR
{
   WORD phywadr[ MACADDRESS_WORDS_48 ];
   inline bool IsNull() {return ((phywadr[2]==0) && (phywadr[1]==0) &&
           (phywadr[0]==0)); };
   inline bool IsMultiCast() { return (phywadr[0]&0x0100);};
   inline bool IsBroadCast() { return ((phywadr[0]==0xFFFF) &&
           (phywadr[1]==0xFFFF) && (phywadr[2]==0xFFFF)); };
   void print();
} __attribute__( ( packed ) ) MACADR;
```

`MACADR` is defined in nburn/include/nettypes.h.

10.3 GetCurrentAP

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
void GetCurrentAP(driverStatusStruct *ap);
```

Description:
Get the full status of the connected network in the form of a driverStatusStruct structure.

Parameters:

ap	A pointer to a driverStatusStruct structure to be populated with the status information of the current WiFi network connection.

Returns:
Nothing.

Notes:
```
struct driverStatusStruct {
    uint8_t     connected;
    uint8_t     ssidLength;
    uint16_t    txPower;
    int16_t     rssi;
    uint8_t     band;
    uint8_t     channel;
    uint32_t    maxTxRate;
    uint8_t     security;
    uint8_t     cipher;
    MACADR      bssid;
    uint8_t     bssType;
    char        ssid[SSID_MAX_LEN + 1]; // Placeholder for char array
                                           pointer
    uint32_t    tickLastUpdated;
};
```

`driverStatusStruct` is defined in nburn/include/nbwifi/nbWifiDriver.h.

10.4 GetDeviceInformation

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
void GetDeviceInformation(nbWifiDeviceInfo *ap);
```

Description:
Get NBWifi module device information in the form of a nbWifiDeviceInfo structure.

Parameters:

ap	A pointer to a nbwifiDeviceInfo structure to be populated with the NBWIFI module device information.

Returns:
Nothing.

Notes:
```
struct nbWifiDeviceInfo {
    uint8_t     hardwareMajorRev;
    uint8_t     hardwareMinorRev;
    uint8_t     softwareMajorRev;
    uint8_t     softwareMinorRev;
    MACADR      hwAddr;
    uint8_t     hardwareTypeLength;
    char        hardwareType[]; // Placeholder for char array pobe_inter
};
```

`nbWifiDeviceInfo` is defined in nburn/include/nbwifi/nbWifiDriver.h.

10.5 Connected

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
`bool Connected();`

Description:
Check if the WiFi interface is currently connected to a network. This function performs a bus communication to query the WiFi module. This will perform more slowly than GetLinkStatus(), but is more reliable of a connection status.

Parameters:
None.

Returns:
TRUE, if connected. FALSE, otherwise.

10.6 GetLinkStatus

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
bool GetLinkStatus();
```

Description:
Get the link status of the WiFi interface. This function does not perform a bus communication, which means it will perform quickly but is potentially less reliable than Connected(). This function returns a maintained connection status variable.

Parameters:
None.

Returns:
Returns TRUE, if connected. Returns FALSE, otherwise.

10.7 GetSignalStrength

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
int GetSignalStrength();
```

Description:
Get the recieved signal strength indicator (RSSI) of the current connection.

Parameters:
None.

Returns:
A value between 0 and -100, if a valid RSSI value is found. The signal strength is greater as the value approaches 0.

10.8 GetCurChannel

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
uint8_t GetCurChannel();
```

Description:
Get the 802.11 channel of the current connection.

Parameters:
None.

Returns:
802.11 channel of the current connection, or 0 if the request failed.

10.9 GetSecurity

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
uint8_t GetSecurity();
```

Description:
Get the security type of the current connection.

Parameters:
None.

Returns:
`SEC_VALUE_OPEN`, `SEC_VALUE_WEP`, `SEC_VALUE_WPA`, `SEC_VALUE_WPA2`, `SEC_VALUE_WPS`, and `SEC_VALUE_UNKNOWN`, which are defined in nburn/include/nbwifi/nbWifiConstants.h.

10.10 GetCipher

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
`uint8_t GetCipher();`

Description:
Get the security cipher of the current connection.

Parameters:
None.

Returns:
`CIPH_VALUE_NONE`, `CIPH_VALUE_TKIP`, `CIPH_VALUE_AES`, `CIPH_VALUE_MIXED`, or `CIPH_VALUE_UNKNOWN`, which are defined in nburn/include/nbwifi/nbWifiConstants.h.

10.11 SetMAC

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
void SetMAC( const MACADR *newMAC, bool waitForResponse = true);
```

Description:
Set the MAC address of the WiFi interface.

Parameters:

newMAC	A pointer to the MACADR structure used by this function to set the MAC address of the WiFi module.
waitForResponse	If TRUE, pend for up to WIFI_PEND_TIMEOUT (10) seconds for the command to be processed by the WiFi module. WIFI_PEND_TIMEOUT is defined in nbwifi/include/nbwifi/nbWifiDriver.h. If FALSE, the function returns immediately after sending the command.

Returns:
Nothing.

Notes:
```
typedef struct MACADR
{
   WORD phywadr[ MACADDRESS_WORDS_48 ];
   inline bool IsNull() {return ((phywadr[2]==0) && (phywadr[1]==0) &&
           (phywadr[0]==0)); };
   inline bool IsMultiCast() { return (phywadr[0]&0x0100);};
   inline bool IsBroadCast() { return ((phywadr[0]==0xFFFF) &&
           (phywadr[1]==0xFFFF) && (phywadr[2]==0xFFFF)); };
   void print();
} __attribute__( ( packed ) ) MACADR;
```

`MACADR` is defined in nburn/include/nettypes.h.

10.12 UpdateSlaveFirmware

Header File:
#include <nbwifi\nbWifiDriver.h>   // Found in C:\nburn\include\nbwifi

Synopsis:
```
bool UpdateSlaveFirmware( uint32_t imageLength, const uint8_t *imageBuffer
);
```

Description:
Update the WiFi module's firmware.

Parameters:

imageLength	Length of the firmware image in bytes.
imageBuffer	A pointer to the buffer containing the new firmware image.

Returns:
Returns, TRUE if successful. Otherwise, returns FALSE.

11. Revision History

Revision	Date	Description
1.0	11/1/2017	Initial release