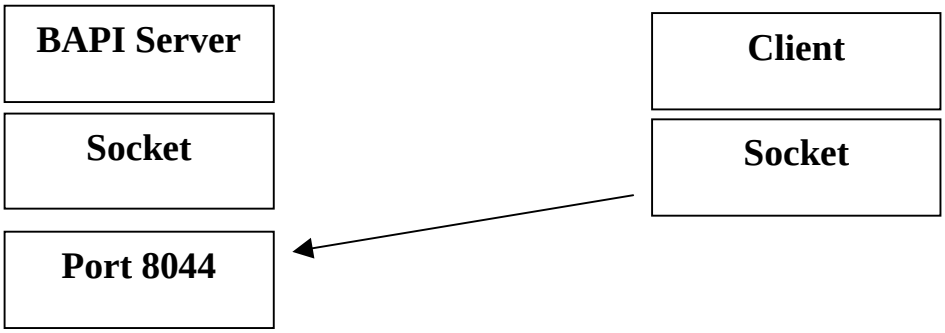
Bitbus Open Technical Meeting

Fiorano Modenese (Italy) - May 5, 2000

A modest proposal for a TCP/IP to Bitbus Gateway

by Mario G. Casali – System Electronics

The gateway is based upon a ***BAPI Server***, which creates a listening socket bound to ***port 8044***.



BAPI Client sends TCP/IP messages to **BAPI Server**, which answers back to the BAPI Client with another TCP/IP message.

Exchanged messages have the following format:

- mandatory message header;
- optional message parameters.

Both the header and the data must have an even number of bytes. All words and double word are stored using little endian (Intel) order.

0000	0x6C	0x1F	**Message Header** Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes for current release of gateway
0004			Size of message parameters: 0 to 1016 *(must be even)*
0006			Function Code *(see table)*
0008	Byte000	Byte001	**Message Parameters**
000A	Byte002	Byte003	Meaning of bytes in this part of the message depends upon Function Code
	. . .		
0008+ nnnn	ByteNNN	Filler (if necessary)	

As one can see from the following table, most Function codes are one-to-one associated to BAPI routines.

Code	Function	Message Parameters
0x9999	Disconnect link	None
0x0001	Call BitbusOpenMaster	AppName *(szString)*
		BitbusDevice *(szString, plus zero/one filler byte)*
		OpenData *(optional, implementation defined structure)*
0x0002	Get value returned by BitbusOpenMaster	hBitbus *(BBHANDLE)*
0x0003	Call BitbusOpenSlave	AppName *(szString)*
		BitbusDevice *(szString, plus zero/one filler byte)*
		TaskId *(BYTE)*
		FunctionId *(BYTE)*
		OpenData *(optional, implementation defined structure)*
0x0004	Get value returned by BitbusOpenSlave	hBitbus *(BBHANDLE)*
0x0005	Call BitbusClose	hBitbus *(BBHANDLE)*
0x0006	Get value returned by BitbusClose	rc *(INT32)*
0x0007	Call BitbusSendMsg	hBitbus *(BBHANDLE)*
		Msg *(used part of BitbusMsg, plus zero/one filler byte)*
0x0008	Get value returned by BitbusSendMsg	rc *(INT32)*
0x0009	Call BitbusWaitMsg	hBitbus *(BBHANDLE)*
		Timeout *(INT32)*
0x000A	Get values returned by BitbusWaitMsg	rc *(INT32)*
		Msg *(used part of BitbusMsg, plus zero/one filler byte)*
0x000B	Call BitbusReset	hBitbus *(BBHANDLE)*
		node *(BYTE, plus one filler byte)*
0x000C	Get value returned by BitbusReset	rc *(INT32)*
0x000D	Call BitbusGetMsgLength	hBitbus *(BBHANDLE)*
		node *(BYTE, plus one filler byte)*
0x000E	Get value returned by BitbusGetMsgLength	MaxMessageLength *(INT32)*

Some example TCPIP-BAPI frames follow.

Function Code 0x0001 – Call BitbusOpenMaster ***(Client → BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	nn		Size of message parameters: nn bytes
0006	0x01	0x00	Function Code 0x0001
0008	'A'	'P'	Application Name = "APPL"
000A	'P'	'L'	
000C	'0x00'	'B'	Bitbus Device = "BBUS1"
000E	'B'	'U'	
0010	'S'	'1'	
0012	'0x00'	'0x00'	*Last byte is a filler*
0014	0x00	0x00	OpenData *(optional and implementation dependent)*
0016	0x00	0x00	

Function Code 0x0002 – Get return value of BitbusOpenMaster ***(Client ← BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	0x04	0x00	Size of message parameters: 4 bytes
0006	0x02	0x00	Function Code 0x0002
0008	0x01	0x00	Handle to Bitbus = 0x00000001L
000A	0x00	0x00	

Function Code 0x0005 – Call BitbusClose ***(Client → BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	0x04	0x00	Size of message parameters: 4 bytes
0006	0x05	0x00	Function Code 0x0005
0008	0x01	0x00	Handle to Bitbus = 0x00000001L
000A	0x00	0x00	

Function Code 0x0006 – Get return value of BitbusClose ***(Client ← BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	0x04	0x00	Size of message parameters: 4 bytes
0006	0x06	0x00	Function Code 0x0006
0008	0x00	0x00	rc = 0 (BAPI_OK)
000A	0x00	0x00	

Function Code 0x0007 – Call BitbusSendMsg ***(Client → BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	12 .. 260		Size of message parameters: 4+len (+1) bytes
0006	0x07	0x00	Function Code 0x0007
0008	0x01	0x00	Handle to Bitbus = 0x00000001L
000A	0x00	0x00	
000C	_res1	_res2	Bitbus Message
000E	len	Flags	
0010	node	src_dest	
0012	com_res	data[0]	
0014	data[1]	data[2]	
	…		
0nnn	data[len-8]	0x00	*If len is odd, last byte is a filler*

Function Code 0x0008 – Get return value of BitbusSendMsg ***(Client ← BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	0x04	0x00	Size of message parameters: 4 bytes
0006	0x08	0x00	Function Code 0x0008
0008	0x00	0x00	rc = 0 (BAPI_OK)
000A	0x00	0x00	

Function Code 0x0009 – Call BitbusWaitMsg ***(Client → BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	0x08	0x00	Size of message parameters: 8 bytes
0006	0x09	0x00	Function Code 0x0009
0008	0x01	0x00	Handle to Bitbus = 0x00000001L
000A	0x00	0x00	
000C	0xFF	0xFF	Timeout = -1 (BAPI_WAIT_FOREVER)
000E	0xFF	0xFF	

Function Code 0x000A – Get return value of BitbusWaitMsg ***(Client ← BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	12 .. 260		Size of message parameters: 4+len (+1) bytes
0006	0x0A	0x00	Function Code 0x000A
0008	0x00	0x00	rc = 0 (BAPI_OK)
000A	0x00	0x00	
000C	_res1	_res2	Bitbus Message
000E	len	Flags	
0010	node	src_dest	
0012	com_res	data[0]	
0014	data[1]	data[2]	
	…		
0nnn	data[len-8]	0x00	*If len is odd, last byte is a filler*

Function Code 0x000B – Call BitbusReset ***(Client → BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	0x06	0x00	Size of message parameters: 6 bytes
0006	0x0B	0x00	Function Code 0x000B
0008	0x01	0x00	Handle to Bitbus = 0x00000001L
000A	0x00	0x00	
000C	node	0x00	Node *(plus filler byte)*

Function Code 0x000C – Get return value of BitbusReset ***(Client ← BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	0x04	0x00	Size of message parameters: 4 bytes
0006	0x0C	0x00	Function Code 0x000C
0008	0x00	0x00	rc = 0 (BAPI_OK)
000A	0x00	0x00	

Function Code 0x000D – Call BitbusGetMsgLength ***(Client → BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	0x06	0x00	Size of message parameters: 6 bytes
0006	0x0D	0x00	Function Code 0x000D
0008	0x01	0x00	Handle to Bitbus = 0x00000001L
000A	0x00	0x00	
000C	node	0x00	Node *(plus filler byte)*

Function Code 0x000E – Get return value of BitbusGetMsgLength ***(Client ← BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	0x04	0x00	Size of message parameters: 4 bytes
0006	0x0E	0x00	Function Code 0x000E
0008	0xFF	0x00	Max message length = 255
000A	0x00	0x00	

Function Code 0x9999 – DisconnectLink ***(Client → BAPI Server)***

0000	0x6C	0x1F	Header key (magic number): 8044 = 0x1F6C
0002	0x08	0x00	Size of header: 8 bytes
0004	0x00	0x00	Size of message parameters: 0 bytes
0006	0x99	0x99	Function Code 0x9999

Necessary upgrade of BAPI interface

It would be beautiful if TCPIP frames are handled inside BAPI routines. To do so, we must permit that strings such as

"www.web-bapiserver.com 8044 BBUS1" or

"factory-bapiserver 8044 BBUS1" or

"10.0.1.142 8044 BBUS1"

are valid Bitbus Device Names, as "BBUS1" or "BBUS2" are.
Of course, the first substring is the name of the machine on which the Bitbus card is actually installed, the second substring is the port number of the socket, and the third subtring is the device name of the Bitbus card inside the server machine.

This way, one could write the following fragment of code:

```
hBitbus = BitbusOpenMaster("My Client Application",
                           "factory-bapiserver 8044 BBUS1",
                           0);

rc = BitbusSendMsg(    hBitbus,
                       &BitbusMsg);

...
```

New BAPI error codes must be defined:

```
#define BAPI_ERR_WINSOCK_NOT_AVAILABLE          -50L
#define BAPI_ERR_CANNOT_RESOLVE_HOSTNAME -51L
#define BAPI_ERR_NO_MORE_SOCKET_RESOURCE -52L
#define BAPI_ERR_CANNOT_CONNECT_TO_SERVER       -53L
…
```