Asynchronous Transaction サンプルプログラム

レジスタの読み込みや，バスリセットで，ノードの認識ができたところで，接続先の IEEE1394 機器から，クワッドレットデータを読み込んでみます．このプログラムで IEEE1394 の，ほぼ全体が見えてきます．本ボードに IEEE1394 機器を接続し，相手の IEEE1394 機器からクワッドレットデータを読み込んでみます．このプログラムで IEEE1394 制御の，ほぼ全体が理解できます．横河電機株式会社の IEEE1394 アナライザを使って動作を検証します．

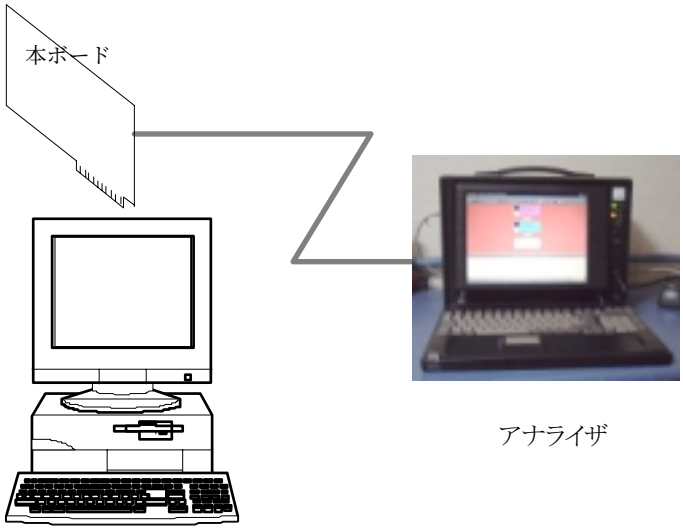

図 接続の様子

読み込み要求について説明します．IEEE1394 で接続された機器のクワッドレットを読み込むには，読み込み要求トランザクションを相手ノードに送らなければなりません．具体的には，クワッドレットデータの読み込みリクエストを，相手先に送信します．read request for quadlet のフォーマットについては，別の資料で説明してありますので，そちらを参照してください．

クワッドレットを読み込むには，IEEE1212 の，どの部分を読み込むかアドレスを指定しなければなりません．これは，コマンド形式で外部から与えるようにしました．以下に，コマンドのシンタックスを示します．

RQ　<IEEE1212 の上位 16 ビットアドレス>　<IEEE1212 の下位 32 ビットアドレス>

IEEE1212 の最上位アドレスの 16 ビット，つまりバス番号と，ノード番号はプログラムが自動的に補います．与えるアドレスは，下位の 48 ビットを 16 ビットと，32 ビットに分けて与えます．

実際にクワッドレットを読み出してみます．プログラムを起動します．プログラムは初期化時にバスリセットをかけます．SelfID が 3 つ読み込まれています．アナライザは IEEE1394 バスをモニタするとともに，コマンドを送ることができます．IEEE1394 は半二重ですので，コマンダとモニタを実装するには2つのノードが必要になります．このため，アナライザは IEEE1394 ノードを2つ持ちます．よって，バスリセットをかけると本ボードが1ノード，アナライザが2ノード，都合3つの SelfID が読み込まれます．

プログラムを起動すると，アナライザはトポロジマップ上に情報を表示しようとし，Configration ROM 周辺へ読み込みトランザクションを送ってきます．今回のプログラムは，これらのトランザクションを無視します．

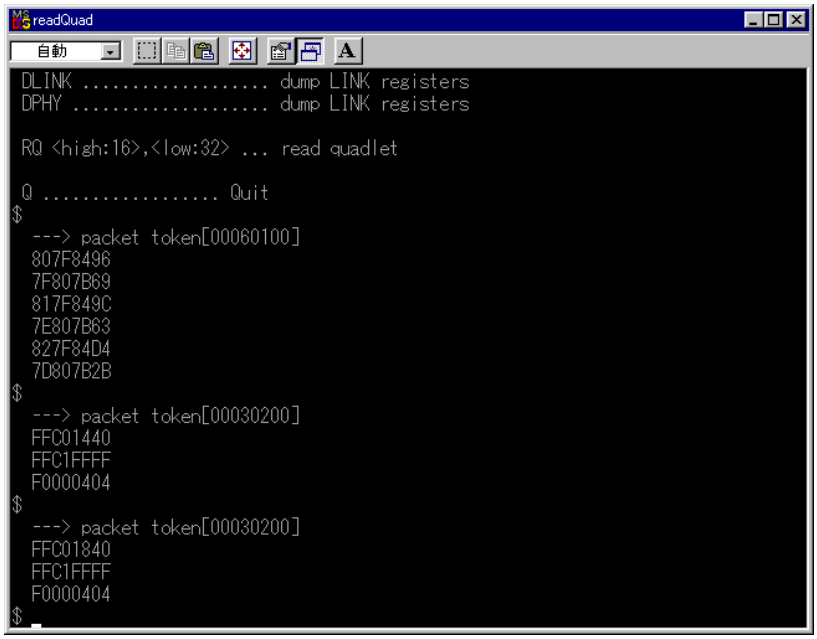

図　アナライザから読み込みコマンドが送られてくる

Configration ROM への読み込み要求を無視したため，アナライザが表示したトポロジマップには「？」が表示されます．各ノードはスピードが 400 と表示されます，これは S400 対応であることを表します．図に現在のトポロジマップが表示されます．アナライザのノードがルートノードになっています．

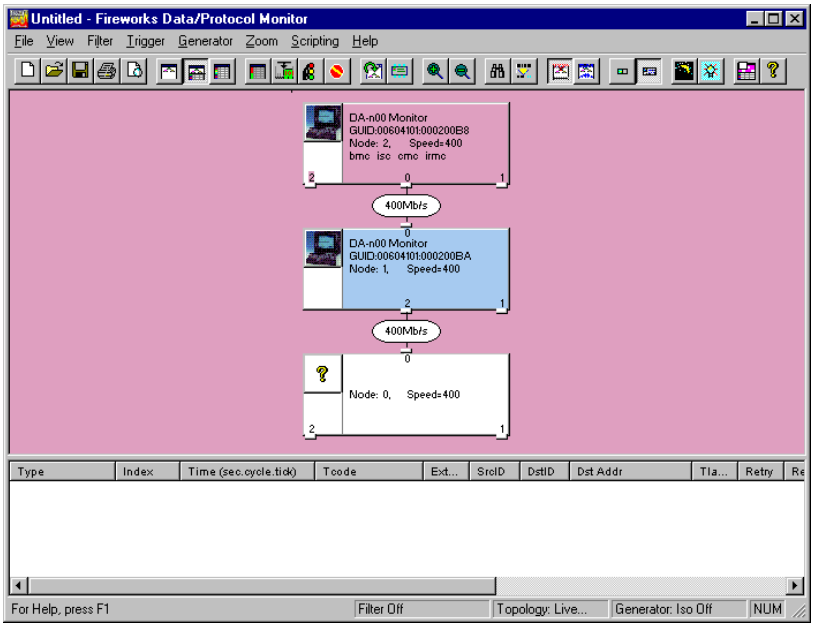

図　トポロジマップ

Read request for quadlet data パケットを送り，IEEE1212 アドレス空間の 0xFFFFF0000404 のクワッドレットを読み込んでみます．アナライザが応答を返してきます．Configration ROM のオフセット 4 には 0x31333934=”1394”が入っています．画面ダンプからわかるように，正常な値が返ってきています．

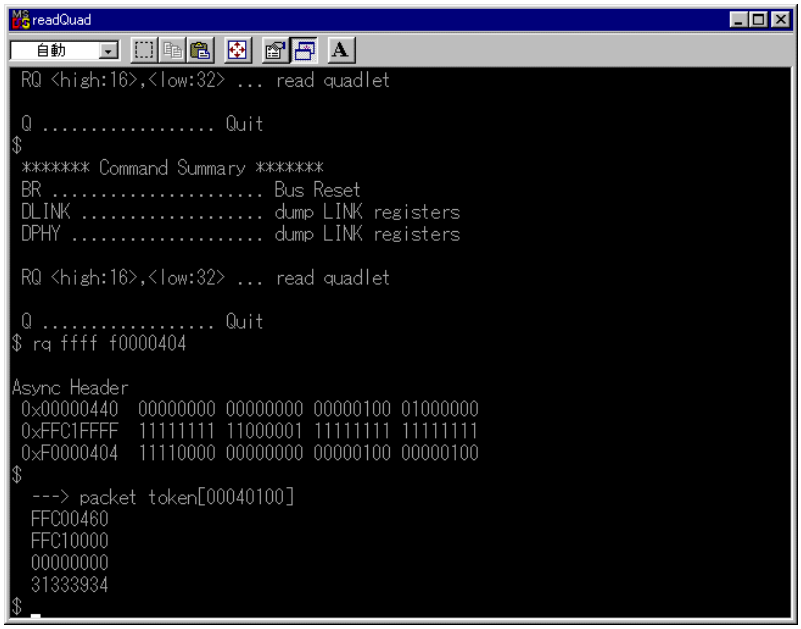

図　Read Quadlet Request パケット送信

アナライザで，当該パケットをキャプチャしてみます．ノード情報や retry コード，CRC などを知ることができます．フォーマットは異なりますが，本ボードが送信時にダンプしたパケットと同じ値になっています．表示はフォーマットされ見やすくなっています．

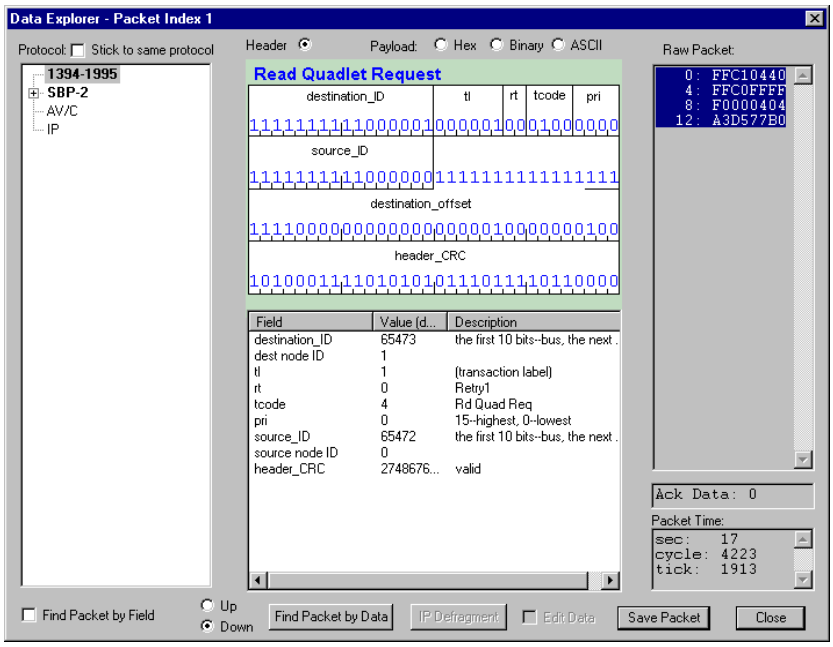

図　Read Quadlet Request パケットキャプチャ

以上，Asynchronous Transaction の様子をアナライザで観測してみました．

本ボードを使えば，アナライザなどが無くてもコマンドを送信したり，アナライザのような動作を行わせることができます．単純な開発キットとしてだけでなく，アナライザとして使用する事ができます．

もう少し高機能なソフトウェアについては，別ドキュメントを参照してください．

プログラム説明

このプログラムは, 5 つのコマンドを受け付けます.

コマンド	説明
BR	LINK レジスタを表示します.
DLINK	LINK レジスタを表示します.
DPHY	PHY レジスタを表示します.
RQ	QUADLET の読み込み要求をターゲットへ送信します. **Syntax：RQ <high:16> <low:32>**
Q	Quit します.

コマンド処理はテーブルドリブンにしてありますので, コマンドを追加するのはテーブルを追加するだけです. Read request for quadlet data パケットは, 4Quadlet で構成されています. しかし, LINK チップへ送るのは 3Quadlet です, 最後の CRC は LINK チップが生成します.

では, 動作の順を追って解説します.

1)tcode に QUADLET 読み込みコードを設定します.

2)スピードは 100M にします, これはテスト対象機器によっては 100M でないと通信できないためです.

3)トランザクションラベルは 1 にしています. IEEE1394 standard を読めば分かりますが, この値はトランザクションごとに, ノード単位で割り振ります. 実際の製品のインプリメントでは, このラベルはノードで管理しなければなりません.

4)IEEE1212 のバス番号をローカルにします.

5)IEEE1212 のノードアドレスを設定します. 自ノードからターゲットノード番号を割り出します.

6)IEEE1212 の下位 48 ビットをパラメータに従い, 設定します.

7)ディバッグのため LINK へ書き込むデータをコンソールへ表示します.

8)ATF へ書き込みます.

これで完了です. この書き込みへ対する, 応答が GRF へ入ってきますが, GRF の読み込みはメインループで行っており, このプロシージャでは行いません.

```
//--------------------------------------------------------------------------
//
// Transmit ASYNC quadlet using info entered by operator
//
void xmitReadRequestPacket( unsigned int highAddr, unsigned int lowAddr )
{
    union async_rdquad_union
    {
        async_rdquad_bit    bits ;
        QUADLET             regs[3] ;
    } async_rdquad ;
    unsigned int index ;

    for( index = 0 ; index < 3 ; index++ )
```

```
        async_rdquad.regs[index] = 0 ;

    async_rdquad.bits.tcode = TCODE_RDQUAD ;
    async_rdquad.bits.speed  = 0 ;  // 100M
    async_rdquad.bits.tl = 1 ;      // every time = 1

    async_rdquad.bits.destBusID = 0x3FF ;
    if( (readPhyReg(MPHY_ID_ADDR) >> 2) == 0 )  // peer to peer
        async_rdquad.bits.destNode = 1 ;
    else
        async_rdquad.bits.destNode = 0 ;

    async_rdquad.bits.destOffsetHigh = highAddr ;
    async_rdquad.bits.destOffsetLow  = lowAddr ;

    printf( "\nAsync Header\n" ) ;
    for( index = 0 ; index < 3 ; index++ ) {
        dumpQuadlet2Bin( async_rdquad.regs[index] ) ;
        printf( "\n" ) ;
    } ;

    putQuadlet2Atf( async_rdquad.regs, 3 ) ;
}
```

より詳しく知りたい場合は, readQuad サンプルを参照してください.