XBus サーボ
プロトコル仕様概略

Ver. 1.0.2

2014 / 6 / 17

JR PROPO 技術開発部

●はじめに

JR PROPO 製 XBus サーボモータ（以下、XBus サーボ）にご興味をお持ちいただき、ありがとうございます。XBus サーボ　プロトコル仕様概略（以下、本ドキュメント）は、XBus サーボを RC 用途以外でもお気軽にお使い頂けるよう、そのプロトコルについて公開し、様々なホビー用途にお使いいただくための便宜を図る目的で作成されました。

今後、様々な形状の XBus サーボがリリースされていく予定ですが、それらは基本的に統一されたプロトコルで動作できるように設計される予定ですので、用途に応じてご選択いただけるものと考えております。

以下の章に記載しました仕様は、まだまだ拡張可能な要素を残しております。もし何かご要望等がございましたら、後述の専用窓口までご相談いただければと思います。

昨今の高性能なマイコン基板を使い、多くの XBus サーボを接続して、様々な作品が生み出されていくことを願っております。

●ドキュメント履歴

リリース日	バージョン	作業者	リリース内容
2014/05/14	1.0.0	澤	初版
2014/05/20	1.0.1	澤	脱字修正、指示値記述追加、コマンドデータパケット説明修正
			XBusコンバーター記述追加、参考例追加
2014/06/17	1.0.2	澤	リミット値ポリシー変更、指示値修正、誤字修正
（以下余白）			

●注意事項

- 本ドキュメントは、XBus サーボのプロトコル仕様概略です。このドキュメントに掲載されていない部分については、今後の拡張の事もあり現時点では全て予約済みとします。

- 本ドキュメントは、製品改良等及びドキュメント改良等のため予告なく変更されることがあります。

- 本ドキュメントは、状況に応じて非公開になる場合があります。その際、下記お問い合わせ窓口も閉鎖になる場合があります。

- 本ドキュメントの無断転載は、最新情報更新の妨げになる場合が多いですのでお控えください。但し、本ドキュメントを基にした解説等の公開や、その際に必要な正しい引用については無論問題ございません。むしろ、歓迎いたします。もしよろしければ、その際は参照用として弊社 web ページへのリンク等を掲示していただければ幸いです。

- 本ドキュメントに関連するトラブル等については、弊社は一切の責任を負わないものとします。なお、製品について修理等が必要な場合には、弊社サービス部宛に通常の修理品としてご依頼頂ければ随時対応いたします。

- 本ドキュメントに関するお問い合わせは、下記の専用メールアドレス宛のみとします。なお弊社サービス部は、本ドキュメントに関しては一切お答えできません。常に XBus サーボ担当者が応対にあたります。また、担当者不在等により頂いたメールの返答には 2 週間程度かかる場合があり、全てのお問い合わせにお答えできるとは限らないものとします。

  専用メールアドレス　　xbusservo@jrpropo.co.jp

●XBus サーボ仕様追補

各 XBus サーボの一般的な製品仕様については弊社 web 及びカタログ等をご覧ください。ここではそれらカタログ等に掲載されていない部分についての説明を行います。

XBus サーボは、電源投入時すぐに内部の初期化を行いますが、以下の条件のどちらかが成立するまでは、モータへ通電しません。

- 正しいシリアルコマンドを受け取り、かつ位置の指示がなされた時
- 正しい RC 系 PWM 信号（以下、PWM 信号）を一定数連続して受け取った時（現時点では 50 パルス以上）

一旦シリアルコマンドを受け取って動作開始した XBus サーボは、電源がオフになるまでシリアルコマンドで動作し、PWM 信号を一切受け取らなくなります。また、一旦 PWM 信号で動作開始した場合は、電源がオフになるまで PWM 信号で動作し、シリアルコマンドを一切受け取らなくなります。

動作開始後、通電したままでシリアルコマンド、もしくは PWM 信号が途切れた場合、約 1.5 秒後に脱力します。なおこの設定は変更でき、脱力しないようにもできます。

●ハードウエアについて

・シリアルバス関連仕様

XBusサーボをマイコン等のホストに搭載されたUART等（以下、ホストUART）で駆動する場合の、ハードウエア条件は以下のとおりです。1線半二重である他は、ごく普通のTTLシリアル信号です。

3.3V TTL 1線　半二重通信（ホストUARTは通常TXモード）
8bit, 1 start bit, 1 stop bit, non parity, LSB first
Idle Hi Level
250kbps BER under 1%

・信号線仕様

信号ケーブルは通常のRC用サーボと同じものであり、各配線の役割も同じです。また、コネクタも通常のRC用と同じものです。

茶	GND
赤	電源（バッテリ公称　4.8V－7.4V）
橙	信号（PWM or XBus）

・諸注意

- パケット送受信に用いるホストUARTの出力には、万が一に備え保護抵抗として50Ωから100Ω程度を信号に直列に入れておくのが望ましいです。ただし、信号線の浮遊容量が大きい場合には、小さめの保護抵抗にしないと波形が歪み、パケット抜けや動作停止（正常なパケットが到着しないことによる脱力等）が発生しますので、ご注意ください。

- 状況によっては、ホストUARTにのみ100kΩ程度のプルアップを設置する方が、動作が安定する場合があります。特に後述のTX、RXの切り替え時にバスが中途半端な電圧になってしまう場合には、プルアップしておくことをお勧めします。

- ホストUARTの信号線に保護抵抗が入っている場合に限り、5V出力のシリアル信号を受け取ることは可能ですが、XBusサーボ側は保護抵抗及び保護ダイオードによるクランプとなるため、あまりお勧めしません。また、その場合でもXBusサーボから返される信号は3.3Vのシリアル信号となります。

- 比較的高速なシリアルバスのため、キャパシタを用いたフィルタ類は一切厳禁とします。オープンドレイン、オープンコレクタの使用も推奨しません。pF単位の容量でも影響を及ぼすことがあるため、配線には注意してください。長距離を伝送する必要がある場合には、適宜双方向バッファ等の設置が望ましいです。

●ソフトウエアについて

・概略

このプロトコルは、基本的にバイト列で形成されたパケットを用いて通信を行うシステムです。パケットの内容により、XBus サーボから応答のパケットが返ってくるものと、返ってこないものがあります。

全てのパケットには、末尾に CRC が設定されており、その正当性が担保されます。XBus サーボ内部においては、CRC エラーが出たパケットは全て破棄されます。また、ホストが受領したパケットについては、ホスト側にて CRC の確認をするのが望ましいです。

各 XBus サーボには後述の方法でチャンネル ID を設定できます。チャンネル ID は、一連の XBus の接続の中では、全てユニークでなければなりません。なお、チャンネル ID の設定は、トラブルを回避する観点で事前に個別にホスト UART に接続して行うのが望ましいです。

パケットには大きく分けて 2 種類あり、XBus サーボの角度指示を行うチャンネルデータパケットと、その他の各種設定を行うコマンドデータパケットがあります。コマンドデータパケットについては、さらにいくつかの種類があります。

通常の使い方としては、ホスト UART は TX モードに設定しておき、PWM 信号のように間欠的に XBus サーボに対してチャンネルデータパケットを送り続ける形になります。XBus サーボは、送り続けられる角度指示に従い続けます。

同一のバスに同一のチャンネル ID を持つ XBus サーボを複数接続することは、プロトコル上問題が出る場合がありますので基本的に禁止していますが、異なるチャンネル ID を持つ XBus サーボを、後述のサブ ID を活用することで 4 個まで連動させることが可能です。

コマンドデータパケットを用いることで、原則として任意のタイミングで任意の XBus サーボに対してその設定を変更することができます（一部例外あり）。また、同じく任意のタイミングで任意のステータスを確認することもできます。ただし、1 線半二重ですので、バスの管理には注意が必要です。

・チャンネルデータパケット

XBusサーボへの角度指示を行います。本パケットのバイト列は以下のとおりです。

バイト列オフセット	名前	定義
0	Command	0xA4
1	Length	2 項から CRC8 の手前までのバイト数
2	Key	0x00
3	Type	0x00
4-0	CH-ID	→解説 1
4-1	CH-Function	0x00
4-2	CH-Data-High	→解説 2
4-3	CH-Data-Low	→解説 2
(4 項　繰り返し)		→解説 3
n	CRC8	→解説 4

なお、本パケットに対するXBusサーボからの応答パケットは発生しないので、ホストUARTはパケット送信後もTX状態のままで構いません。

基本的には、接続されている全てのXBusサーボに対する角度指示を一つのパケットに内包し、一括送信する形になります。しかし、XBusサーボは本パケット内に自身宛のデータが無くてもエラーとは判定しませんので、例えば複数パケットに別々のチャンネルIDの角度指示を載せる等して、分割して送信することも可能です。

解説 1　チャンネル ID について

各XBusサーボには、それぞれにユニークな8bitのチャンネルIDが割り振られますが、これは以下のような構造になっています。

チャンネル ID の構造

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
サブ ID (0,1,2,3)		サーボID (1 - 50)					

チャンネルデータパケットで指定できるのはサーボIDのみであり、サブIDは0を指定します。チャンネルデータパケットを受け取ったXBusサーボは、XBusサーボ自身のサブIDに何番が指定されていようと、サーボIDが一致する指示に従います。

後述する個々のXBusサーボの各種設定については、サブIDを含めたチャンネルIDの一致で動作するため、これにより最大4個までのXBusサーボが、同一のサーボIDによって同期的に動作することが可能になります。

製品出荷時のチャンネルIDは0x01です。

解説２　指示値について

High Lowの2byteで位置指示を行います。この値は、従来のPWM信号の概念を踏襲しており、以下のような値で指示します。XBusサーボは概ね850-2150uSecの範囲で動作するようになっています。それ以外の範囲での角度指定については、脱力します。

0x0000	800uSec	
0x1249	900uSec	（-60度、または-90度）
0x7FFF	1500uSec	（0度）
0xEDB6	2100uSec	（+60度、または+90度）
0xFFFF	2200uSec	

ただし、後述のリミット値が設定されている場合、その値よりも外側の指示についてはリミット値でクリップされるため、設定によっては脱力せず、リミット位置で停止します。現時点では、リミット値が工場出荷時に 800－2200 で設定されているため、リミット値は無効になっています。

解説３　繰り返しについて

本パケットは不定長であり、ユーザが必要とするチャンネル数分のデータを内包した形で１個のパケットとして送信できます。ただし、同じサーボIDは複数指定できないため、サーボIDの最大数（50個分）が限度となります。

繰り返す場合は4項の4バイト分を一塊として繰り返す形になりますので、それぞれチャンネルIDと指示値が繰り返し表記される形になります。

本パケット内では、チャンネルIDは連続していてもいなくても構いませんし、チャンネルIDの順番にも特に制限はありません。

＜＜　一例　＞＞

Cmd	Len	Key	Type	CH-ID	CH-F	CH-D-H	CH-D-L	CH-ID	CH-F	CH-D-H	CH-D-L	CRC
0xA4	0x0A	0x00	0x00	0x01	0x00	0xnn	0xnn	0x03	0x00	0xjj	0xjj	0xkk

基本的にXBusサーボはパケットの先頭から順番に自身のサーボIDを探しますので、長いパケットの場合は最初と最後でミリセカンドレベルのタイミング差が出る可能性があります。

XBusサーボ自身が持つバッファサイズを超えるパケットが送信された場合、たとえCRCが正常であってもエラーになります。その結果、当該パケットは無視され次のパケットを待つモードに戻ります。その際なんらかのステータスが返送されるわけではありませんのでご注意ください。

解説４　CRC8 について

本パケットに適用されるCRC式は以下のとおりです。CRCの範囲は、パケットの先頭からCRC手前までのバイト全部です。本ドキュメント末尾にサンプルソースを掲載しましたので、参考になさってください。

```
X8 + X5 + X4 + 1
```

全てのパケットにおいてCRCのチェックを行いますので、ホスト側のコードを書く上では重要なポイントの一つになります。

・コマンドデータパケット

XBusサーボへの各種設定を行います。本パケットのバイト列は以下のとおりです。

バイト列	名前	定義
0	Command	→解説１
1	Length	2 項から CRC8 の手前までのバイト数
2	Key	0x00
3	CH-ID	→解説２
4	Order	→解説３
5-0	Data1	→解説３
5-1	Data2	→解説３
6	CRC8	→解説４

なお、本パケットに対しては、後述する一部の例外を除き、基本的にXBusサーボから後述のStatus Commandを用いた応答パケットが発生しますので、ホストUARTはパケット送信完了後速やか（数uSec程度）にTXを開放してRXに切り替え、Status Commandパケット受信の準備をしてください。

ただし、ホストUART内部に送信バッファを持つ場合、確実にパケットのバイト列が送信完了していることを確認してから切り替えを行わないと、XBusサーボがパケットを全て受信できず、エラーになる場合があります。

現時点では明確なタイムアウトは設定されていませんが、遅くとも14mSec以内に何らかの反応がなければ、失敗したものと見做して構いません。

解説１　コマンドについて

本パケットは、以下の３つのコマンドに適用されます。

0x20	Set Command	後述するオーダーに沿った値を設定するコマンド
0x21	Get Command	後述するオーダーに沿った値を読み出すコマンド
0x22	Status Command	応答に使われるコマンド（ユーザ使用不可）

ホストがSet Command、またはGet Commandを送信すると、XBusサーボは後述するオーダーに沿った作業を行い、その結果として同じオーダー、同じバイト数のStatus Commandを応答してきます。

XBusサーボから送信されたStatus Commandは、通常当該オーダーに関する現状の値を返してきます。Set Commandに対する応答の場合、状況によっては、設定しようとした値ではなくXBusサーボ側での限界値にクリップされた値で帰ってくることがあります。

XBusサーボが対応できないオーダーだった場合、後述のUnsupported Orderに変更されたStatus Commandが返ってきます。この場合、Data2は存在しませんので、元のオーダーによってはバイト数が１バイト少ないサイズで返ってくることに注意してください。

このように、Status Commandのパケットサイズは変化する可能性がありますので、必ず受信中の２バイト目で判断して受信長を決めるようにしてください。

解説２　チャンネル ID について

前項のチャンネルIDと同仕様ですが、ここではサブIDも適用されます。従って、最大200個のXBusサーボを相手にできますが、本コマンドは応答が前提であることもあり、1個のXBusサーボにつき毎回1パケットを必要とします。

また、Set Commandついては、ここに0x00を指定するとすべてのチャンネルIDを持つXBusサーボに対して効力を持ちます。このケースに限っては、Status Commandを用いた応答パケットは発生しません。なお、誤ってGet Command に0x00を指定してしまっても、応答はありませんのでご注意ください。

ご注意

複数のXBusサーボが接続された状態で、チャンネルIDに0x00を指定したID変更やリセット関係のオーダーを実行すると、全てのXBusサーボが同じチャンネルIDになってしまいます。その後のチャンネルIDを指定したコマンドデータパケットに対しては、それぞれがほぼ同時に応答してしまってバスが混乱するので注意してください。

解説３　オーダー、データについて

オーダー、データに関する役割等は以下のとおりです。本パケットでは、オーダーによってはData2が存在せず、パケット全体のバイト数も変化することに注意してください。2バイトのデータについては、特に断りがない限りData1が上位バイト、Data2が下位バイトとなります。また、基本的に符号付データとして扱います。

一部のオーダーを除き、基本的に全てのオーダーは後述のOperate Modeにて動作し、指示に成功すれば即座に動作に反映されます。

基本的に、Parameter Writeコマンドを送信しない限り、パラメータはROM領域には記録されません。但し、後述のようにいくつかの動作については例外的に即時書き込まれるので注意してください。

なお、XBusサーボは通常のPWM信号でも動作しますが、その際以下のオーダーにて設定された内容は反映されません。

Reverse、Neutral、Travel High、Travel Low、Limit High、Limit Low

Order	名前	data	意味	初期値	備考
0x01	Mode	1	XBus サーボのモード切替を行う。現時点で有効なモードは以下のとおり 0x01 Operate Mode 0x02 ID Setting Mode	0x01	
0x03	ID	1	チャンネル ID を指示した値に変更する。Mode Order にて ID Setting Mode に切替えた時のみ有効となる。 本オーダー実行後は、自動的に Operate Mode へ移行し、新しいチャンネル ID で動作するとともに、ROM 領域へ自動で記録される。 CH-ID を 0x00 に設定することで、不明なチャンネル ID を持つ XBus サーボでも設定できる。	0x01	
0x04	Version	2	ファームウエアバージョンを返す。	機種毎	get only
0x05	Product	2	機種番号（別表２参照）	機種毎	get only

0x06	Unsupported	1	Status Command 専用のオーダー。ホストが送信したオーダーを受け付ける事が出来ない場合に XBus サーボから返ってくる。 この際、データには対応できなかったオーダーが返される。	------	don't use
0x07	Parameter Reset	2	下記別表 1 の Index で指定したパラメータを初期値に戻す。	------	set only
0x08	Parameter Write	2	下記別表 1 の Index で指定したパラメータを ROM 領域へ書き込む。	------	set only
0x10	Reverse	2	0x0000 通常動作 0x0001 左右反転動作	0x0000	
0x11	Neutral	2	ニュートラル位置（1500uSec）に対するオフセットを指定する。±600 の範囲で指定できる。主に複数 XBus サーボの同期連動時補正に使用する。	0	
0x12	Travel High	2	ニュートラル位置（1500uSec）よりも上の領域において、角度指示を拡大する。通常 128、最大 192 まで指定できる。主に複数 XBus サーボの同期連動時補正に使用する。	128	
0x13	Travel Low	2	ニュートラル位置（1500uSec）よりも下の領域において、角度指示を拡大する。通常 128、最大 192 まで指定できる。主に複数 XBus サーボの同期連動時補正に使用する。	128	
0x14	Limit High	2	角度指示できる最大値。Limit Low 未満には設定できない。無理に設定しても、自動的に Limit Low に修正される。	0xFFFF	
0x15	Limit Low	2	角度指示できる最小値。Limit High より大きくは設定できない。無理に設定しても、自動的に Limit High に修正される。	0x0000	
0x16	P Gain	1	XBus サーボが持つ P ゲインに対する増減値。±50 の範囲で指定できる。	0	
0x17	I Gain	1	XBus サーボが持つ I ゲインに対する増減値。±50 の範囲で指定できる。	0	
0x18	D Gain	1	XBus サーボが持つ D ゲインに対する増減値。±50 の範囲で指定できる。	0	

0x19	Dead Band	1	XBus サーボが持つデッドバンドに対する増減値。±10 の範囲で指定できる。	0	
0x1A	Boost	2	XBus サーボが持つブースト値に対する増減値。±999 の範囲で指定できる。	0	
0x1B	Alarm Level	1	XBus サーボからアラームが発せられるパワー閾値。0-100%で指定できる。アラームが発せられると、モータ音が変化する。	機種毎	
0x1C	Alarm Delay	2	XBus サーボからアラームが発せられるまでの遅延時間を指定できる。この時間が経過しない範囲で閾値から下がれば、警告されない。0-5000mSec まで指定できる。	機種毎	
0x1D	Angle	1	0x00 通常動作（最大角 120 度） 0x01 最大角 180 度（一部の機種では、180 度に到達しない場合がある）	機種毎	
0x1E	Slow Start	1	0x00 起動後、すぐに通常動作する。 0x01 起動時、最初に取り込んだ指示位置までゆっくり移動する。ただし、移動中に指示位置が変化した場合、その時点から通常動作に戻る。	機種毎	
0x1F	Stop Mode	1	0x00 通常動作（角度指示が途絶えると 1 秒前後で脱力する） 0x01 ホールド動作（角度指示が途絶えると直前の位置を維持する）	0x00	
0x20	Current Position	2	現在の出力軸の位置を返す。静止していても、指示位置と一致しているとは限らないので注意すること。	------	get only
0x21	Current Power	1	現在モータへ掛けているパワーを返す。0-100%の値を示す。	------	get only
0x22	Speed Limit	1	0 通常動作 1-30 速度制限モード（1 が最も遅い）	機種毎	
0x23	Max Integer	2	I ゲインの積分値リミッタに対する増減値。±999 の範囲で指定できる。	0	

別表１　Parameter Reset、Parameter WriteにおけるIndex

Index	名前	適用	備考
0x0001	All Data with ID	reset only	CH-ID を 0x00 にする必要がある。ROM 領域へ自動で記録される。リセットに成功すると、チャンネル ID はデフォルトに戻る。
0x0002	All Data without ID	reset only	
0x0003	Servo ID	reset only	CH-ID を 0x00 にする必要がある。ROM 領域へ自動で記録される。リセットに成功すると、チャンネル ID はデフォルトに戻る。
0x0004	Reverse	both	
0x0005	Neutral	both	
0x0006	Travel High	both	
0x0007	Travel Low	both	
0x0008	Limit High	both	
0x0009	Limit Low	both	
0x000A	P gain	both	
0x000B	I gain	both	
0x000C	D Gain	both	
0x000D	Dead Band	both	
0x000E	Boost	both	
0x000F	Alarm Level	both	
0x0010	Alarm Delay	both	
0x0011	Angle	both	
0x0012	Slow Start	both	
0x0013	Stop Mode	both	
0x0014	Speed Limit	both	
0x0015	Max Integer	both	

別表２　Productにおける応答値／機種名一覧

応答値	機種名
0x0200	NX8921
0x0201	NX3421
0x0202	NX588

解説４　CRC8 について

本パケットに適用されるCRC式は、前項と同じく以下のとおりです。CRCの範囲は、パケットの先頭からCRC手前までのバイト全部です。本ドキュメント末尾にサンプルソースを掲載しましたので、参考になさってください。

```
X8 + X5 + X4 + 1
```

全てのパケットにおいてCRCのチェックを行いますので、ホスト側のコードを書く上では重要なポイントの一つになります。

・パケット送信間隔について

通常、XBus用RC受信機においてはこれらのパケットを14mSec間隔で送出しています。原則としてパケットの送信間隔はこれが標準ですが、XBusサーボの実力値としては、さらに間隔を詰めた送信を行うことが可能です。

ただし、どこまで詰めることができるのかについては機種によって異なる可能性がありますので、将来に渡って保障可能な間隔も14mSecとします。

この問題は、特にチャンネルデータパケットにおいて重要になります。XBusサーボに高速な反応を期待する場合、指示値を出力するペースを速くする必要があるからです。

何チャンネル分のデータを一つのパケットに載せて送信するのかによっても、どこまで詰めることができるのかは変化します。チャンネル数が少なければ、パケット一つに必要な処理時間が短くなるからです。

パケット送信間隔を短くしすぎた場合、先行パケットの処理が終わらないうちに後続パケットが到着してしまう可能性があります。この場合、後続パケットはロストすることになりますが、再度パケット取得に復帰するには、ロストパケット送信終了後、最低でも１バイト分以上のアイドル時間が必要となる事に注意してください。

参考例ですが、現行機種で実験した場合、４チャンネル分のデータを載せたパケットで、1.5mSec程度の送信ペースで問題なく動作した実例があります。この場合、パケットの理論的な時間は0.84mSecになり、パケット間隔は0.66mSecになります。

同様に50個分のデータを載せたチャンネルデータパケットの時間は、理論値で8.2mSecとなります。仮に60フレーム／秒で動作させた場合、フレームレートは16.67mSecとなりますので、一般的なモーション動作には充分に間に合う計算になります。

ただし、これらの理論値はあくまでホストUARTからの送信がビット列的に最密で送信された場合であり、ホストUARTのプログラミングによっては、バイト間に隙間ができてしまう等、所定の性能が出ない可能性がある事に注意してください。

なお、現行機種のXBusサーボ制御周期は1mSecですので、1mSecよりも短縮することには意味がありません。但し、将来の機種においてはこの限りではありません。

●参考

・CRC8 算出用サンプルソース

```
static uint8_t s_crc_array[256] =
{
    0x00, 0x5e, 0xbc, 0xe2, 0x61, 0x3f, 0xdd, 0x83,
    0xc2, 0x9c, 0x7e, 0x20, 0xa3, 0xfd, 0x1f, 0x41,
    0x9d, 0xc3, 0x21, 0x7f, 0xfc, 0xa2, 0x40, 0x1e,
    0x5f, 0x01, 0xe3, 0xbd, 0x3e, 0x60, 0x82, 0xdc,
    0x23, 0x7d, 0x9f, 0xc1, 0x42, 0x1c, 0xfe, 0xa0,
    0xe1, 0xbf, 0x5d, 0x03, 0x80, 0xde, 0x3c, 0x62,
    0xbe, 0xe0, 0x02, 0x5c, 0xdf, 0x81, 0x63, 0x3d,
    0x7c, 0x22, 0xc0, 0x9e, 0x1d, 0x43, 0xa1, 0xff,
    0x46, 0x18, 0xfa, 0xa4, 0x27, 0x79, 0x9b, 0xc5,
    0x84, 0xda, 0x38, 0x66, 0xe5, 0xbb, 0x59, 0x07,
    0xdb, 0x85, 0x67, 0x39, 0xba, 0xe4, 0x06, 0x58,
    0x19, 0x47, 0xa5, 0xfb, 0x78, 0x26, 0xc4, 0x9a,
    0x65, 0x3b, 0xd9, 0x87, 0x04, 0x5a, 0xb8, 0xe6,
    0xa7, 0xf9, 0x1b, 0x45, 0xc6, 0x98, 0x7a, 0x24,
    0xf8, 0xa6, 0x44, 0x1a, 0x99, 0xc7, 0x25, 0x7b,
    0x3a, 0x64, 0x86, 0xd8, 0x5b, 0x05, 0xe7, 0xb9,
    0x8c, 0xd2, 0x30, 0x6e, 0xed, 0xb3, 0x51, 0x0f,
    0x4e, 0x10, 0xf2, 0xac, 0x2f, 0x71, 0x93, 0xcd,
    0x11, 0x4f, 0xad, 0xf3, 0x70, 0x2e, 0xcc, 0x92,
    0xd3, 0x8d, 0x6f, 0x31, 0xb2, 0xec, 0x0e, 0x50,
    0xaf, 0xf1, 0x13, 0x4d, 0xce, 0x90, 0x72, 0x2c,
    0x6d, 0x33, 0xd1, 0x8f, 0x0c, 0x52, 0xb0, 0xee,
    0x32, 0x6c, 0x8e, 0xd0, 0x53, 0x0d, 0xef, 0xb1,
    0xf0, 0xae, 0x4c, 0x12, 0x91, 0xcf, 0x2d, 0x73,
    0xca, 0x94, 0x76, 0x28, 0xab, 0xf5, 0x17, 0x49,
    0x08, 0x56, 0xb4, 0xea, 0x69, 0x37, 0xd5, 0x8b,
    0x57, 0x09, 0xeb, 0xb5, 0x36, 0x68, 0x8a, 0xd4,
    0x95, 0xcb, 0x29, 0x77, 0xf4, 0xaa, 0x48, 0x16,
    0xe9, 0xb7, 0x55, 0x0b, 0x88, 0xd6, 0x34, 0x6a,
    0x2b, 0x75, 0x97, 0xc9, 0x4a, 0x14, 0xf6, 0xa8,
    0x74, 0x2a, 0xc8, 0x96, 0x15, 0x4b, 0xa9, 0xf7,
    0xb6, 0xe8, 0x0a, 0x54, 0xd7, 0x89, 0x6b, 0x35,
};

static uint8_t crc_table(uint8_t data, uint8_t crc)
{
    uint16_t index = (data ^ crc) & 0xff;
    crc = s_crc_array[index];
    return crc;
}

uint8_t crc8(uint8_t * buffer, uint8_t length)
{
    uint8_t crc = 0;

    while (length-- > 0)
            crc = crc_table(*buffer++, crc);
    return crc;
}
```

・チャンネル ID の変更方法について

チャンネル ID の変更については、不慮のトラブルをできるだけ避けるため、以下の手順でないと変更されないようになっています。単純に ID オーダーを送信しただけでは変更されませんので、ご注意ください。

1. XBus サーボへ Mode オーダーを送信し、ID Setting Mode に移行させる
2. XBus サーボへ ID オーダーを送信し、ID を変更する（この段階で自動的に元のモードに戻る）

・XBus コンバーターについて

XBus サーボに混在して従来の PWM サーボを使用したい場合、弊社ではコンバーターをご用意しております。ただし、コンバーターについては対応する機能が限定されており、以下の機能のみ作動します。

- チャンネルデータパケットの全機能（但し、16 チャンネル分まで）
- コマンドデータパケットの以下のオーダー
  - `Mode、ID、Version、Unsupported、Parameter Reset、Parameter Write、Reverse、Neutral、Travel High、Travel Low`

また、4 ポートのコンバーターの場合、初期状態やチャンネル ID のリセット等を行った直後は、サーボ ID に全て 1、サブ ID にそれぞれ 0 から 3 の値がセットされます。

なお、コンバーターが出力する PWM 波形は 20mSec 間隔で固定されているため、それより高速にパケットを送信しても対応できません。速くても 14mSec 間隔程度でのご使用をお願いします。

・同一関節を複数の XBus サーボを用いて構成する場合

二足歩行ロボットの一部でよく見られるリンク形状の脚（以下、リンク脚）を構築する場合や肩の ROLL 軸等、複数のサーボを連動させてトルクアップを図るケースがありますが、この場合、XBus サーボを用いると非常に簡単に設定できます。

XBus サーボでは、サブ ID を用いて最大４個までの XBus サーボを連動させることができますので、たとえばリンク脚の各軸にそれぞれ同一サーボ ID で異なるサブ ID の XBus サーボを装備する方法がお勧めです。

フレーム組み立て中、各 XBus サーボにリバース、ニュートラル位置、最大角（トラベル）を設定し、個々の XBus サーボのズレ等を調整すれば、概ね同期して動作することになります。この調整値は、書き込みオーダーさえ送ってしまえば個々の XBus サーボが内部に記録し、電源を切っても覚えていますので、一度設定してしまえば、あとはメンテナンス等で再調整する以外は再設定の必要はありません。

モーション送信時は、当該サーボ ID＋サブ ID をゼロにした単一のチャンネル ID にてデータを送れば良く、同一サーボ ID の最大４個ともまとめて駆動することが可能です。ですので、チャンネルデータパケット生成の負担や、モーション作成時のデータ的な負担を軽減することが可能になります。

・ホスト UART の TX のみでの使用について

本来、XBus サーボは１線半二重でのやり取りを前提に設計されています。そのため、ホスト UART には半二重通信のための送受信切り替えが必要であり、場合によっては外部回路が必要な場合があります。しかし、以下の条件に合致すれば、ホスト UART の RX を使用せず、切り替え回路も不要で、TX のみで動作させることが可能です。

- 動作中、XBus サーボの状況を一切確認しない。
- XBus サーボへの設定を行うときは、個別に接続して行う。
- 複数の XBus サーボが接続された状態では、一切コマンドデータパケットを送信しない。

つまり、事前に全ての XBus サーボの設定を済ませておきさえすれば、チャンネルデータパケット自身は RX を使用しないので、TX のみで動作させられる事になります。簡易的に動作させたい場合は、こちらがお勧めです。

・XBus サーボに使用しているマイコンについて

現時点では、XBus サーボには Freescale 社の MKL16 シリーズを 48MHz で駆動して使用しています。従いまして、現時点での各ポートの厳密な電圧レベル等は、当該マニュアルの参照をお願いします。但し、今後の製品についてはその限りではありませんので、あまり厳密な電圧レベルに依存した使い方はお勧めしません。

MKL16 マイコンのファームにはセキュリティが掛かっており、無闇に読み出そうとすると、チップの仕様により内部が全てクリアされますのでご注意ください。クリアされた場合、一切動作しなくなります。復活させるにはファームウエアの再書き込み等、修理が必要になります。修理の際は、クリアなさった旨のご申告がないと、基板異常と判断の上、基板そのものの交換作業が発生し高額な修理になる可能性がありますので、ご注意ください。

参考までに、マイコンの信号入力部分の回路図を掲載します。