Future Versatile Group

Global Versatile Controller:GVC

GVC Protocol Handbook

汎用制御装置

GVCプロトコルハンドブック

※本マニュアルに記載されている内容は適時改変される可能性があります。
必ず最新版を確認していただきますようお願い致します。

注意

本書に記載されているデバイスやアプリケーションなどに関する情報は、使用者の便宜のためにのみ提供されているものであり、更新によって無効とされることがあります。アプリケーションと仕様の整合性を保証することは、使用者の責任において行ってください。Future Versatile Groupは、明示的、暗黙的、書面、口頭、法定のいずれであるかを問わず、本書に記載されている情報に関して、状態、品質、性能、商品性、特定目的への適合性をはじめとする、いかなる類の表明も保証も行いません。

Future Versatile Groupは、本書の情報およびその使用に起因する一切の責任を否認します。 Global Versatile Controller(GVC)を生命維持および/ または保安のアプリケーションに使用することはGVCを実際に使用する者(GVC使用者)が全責任において行うものとし、GVC使用者は、デバイスの使用に起因するすべての損害、請求、訴訟、および出費に関してFuture Versatile Groupを弁護、免責し、Future Versatile Groupに不利益が及ばないようにすることに同意するものとします。

暗黙的あるいは明示的を問わず、Future Versatile Groupが知的財産権を保有しているライセンスは一切譲渡されません。

また本書に記載されている商標はその商標の所有者に帰属します。

ライセンス

本書中、またはFuture Versatile Groupが公開している各種プログラムについてはnew BSD license扱いとします。

お願い

Global Versatile Controller:GVCは「汎用制御装置」と言う意味で、ハードに弱い人＆ソフトに弱い人それぞれが簡単に自分のウィークポイントを補完しつつ、ハードとソフトをもっと簡単に繋げたシステムを作ることが出来るように、という目的で提案するものです。PICなどのマスターコントローラー本体と、それらに接続されたセンサーやスイッチなどのモジュール、それらとパソコンやサーバーなどがやり取りをするために定義するプロトコル、そしてそれを実際に処理するデーモンや、各種制御命令を出すCLIコマンドなどの総称です。

昨今PICやAVRなどを使ったマイコンボードが市場に多く登場し、どんどんハード側の敷居が下がってきています。またそれらと組み合わせることを想定した新しいデバイスが日々登場しています。そしてそれらの開発に使うソフトウェア側の環境もマシン語からC言語、そして各種スクリプト言語へとどんどん敷居が下がってきています。

ですが、どうもやっぱりハードとソフト両者の壁はなかなか高くて厚い。その壁を少しでも越えやすくして、双方が出来るだけ簡単に繋がるためにはどうしたらよいか？というかそうするための便利なツールがないかな、と思って探したものの見当たらなかったので、ならば自分たちでつくるか、ということで開発を始めたのがGVCです。

ハードとソフトはそれぞれエキスパートな皆さんにお任せしつつ、本書ではその両者が繋がるために必要な「プロトコル：手順」、つまり橋渡し的な役割をするものについて解説をしていきたいとおもいます。もちろんまだまだ足りない部分ばかりかとは思いますが、これからどんどん充実させていきたいと思いますので、皆さんのご指摘、ご提案、ご協力をいただければ幸いです。

Global Versatile Controller … http://www.GVC-On.net/

目次

4.0x17 (TBD)
4.0x18 (TBD)
4.0x19 (TBD)
4.0x1a (TBD)
4.0x1b (TBD)
4.0x1c (TBD)
4.0x1d (TBD)
4.0x1e (TBD)
4.0x1f (TBD)
4.0x20 (TBD)
4.0x21 GVC_MSG_OTHER : START, RESET, INFO, etc
4.0x22 (TBD)
4.0x23 (TBD)
4.0x24 (TBD)
4.0x25 (TBD)
4.0x26 (TBD)
4.0x27 (TBD)
4.0x28 (TBD)
4.0x29 (TBD)
4.0x2a (TBD)
4.0x2b (TBD)
4.0x2c (TBD)
4.0x2d (TBD)
4.0x2e (TBD)
4.0x2f (TBD)
4.0x30 (TBD)
4.0x31 (TBD)
4.0x32 (TBD)
4.0x33 (TBD)
4.0x34 (TBD)
4.0x35 (TBD)
4.0x36 (TBD)
4.0x37 (TBD)
4.0x38 (TBD)
4.0x39 (TBD)
4.0x3a (TBD)
4.0x3b (TBD)
4.0x3c (TBD)
4.0x3d (TBD)
4.0x3e (TBD)
4.0x3f (TBD)
4.0x40 (TBD)
4.0x41 GVC_MSG_ACCE : 'A' Acceleration (TBD) 加速度
4.0x42 (TBD)
4.0x43 GVC_MSG_COMP: 'C' Compass (TBD) 方位

4.0x44 GVC_MSG_DIST: 'D' Distance (TBD) 距離
4.0x45 GVC_MSG_EARTH: 'E' Earth (TBD) GPS、地図
4.0x46 (TBD)
4.0x47 GVC_MSG_GYRO : 'G' Gyro (TBD) ジャイロ
4.0x48 GVC_MSG_HUMI : 'H' Humidity (TBD) 湿度
4.0x49 GVC_MSG_IR : 'I' Infrared (TBD) 赤外線(焦電)
4.0x4a GVC_MSG_JOY : 'J' Joystick (TBD) ジョイスティック、回転角
4.0x4b (TBD)
4.0x4c GVC_MSG_LIGHT : 'L' LIGHT (TBD) 照度
4.0x4d GVC_MSG_MAG : 'M' Magnetic (TBD) 磁力、磁気
4.0x4e (TBD)
4.0x4f (TBD)
4.0x50 GVC_MSG_PRES : 'P' Pressure (TBD) 圧力
4.0x51 (TBD)
4.0x52 GVC_MSG_RAD : 'R' Radiation (TBD) 放射線
4.0x53 GVC_MSG_SIGHT : 'S' Sight (TBD) 視覚、色
4.0x54 GVC_MSG_TEMP : 'T' Temperature (TBD) 温度
4.0x55 GVC_MSG_UV : 'U' UV (TBD) 紫外線
4.0x56 (TBD)
4.0x57 GVC_MSG_IR : 'W' Wave (TBD) 音波
4.0x58 (TBD)
4.0x59 (TBD)
4.0x5a (TBD)
4.0x5b (TBD)
4.0x5c (TBD)
4.0x5d (TBD)
4.0x5e (TBD)
4.0x5f (TBD)
4.0x60 (TBD)
4.0x61 (TBD)
4.0x62 (TBD)
4.0x63 (TBD)
4.0x64 (TBD)
4.0x65 (TBD)
4.0x66 (TBD)
4.0x67 (TBD)
4.0x68 (TBD)
4.0x69 (TBD)
4.0x6a (TBD)
4.0x6b (TBD)
4.0x6c (TBD)
4.0x6d (TBD)
4.0x6e (TBD)
4.0x6f (TBD)
4.0x70 (TBD)

1.0 概要

Global Versatile Controller:GVCは、PICなどを使用したマスターコントローラー本体と、それらに接続されたセンサーやスイッチなどのモジュール、それらとパソコンやサーバーなどがやり取りをするために定義するプロトコル、そしてそれを実際に処理するデーモンや、各種制御命令を出すコマンドラインコマンドなどの総称です。

モジュールからのデータの取得の仕方やそのデータフォーマット、またモジュールへの命令などプロトコルをきちんと定義することで、ハードウェアとソフトウェアの間の見えない垣根をできるだけ低くして、誰もが簡単に「物理的な装置を含めた統合的なシステム」(フィジカルコンピューティングシステム!?)を構築できるようにという目的の元、各種プログラムソースや本ドキュメント、そしてハードを含めて公開(オープンソース、オープンハードと)します。

GVCの特徴は以下の通りです。

- **汎用制御装置として、各種データの取得や機器の操作が可能**
- **メッセージやコマンドは最大100タイプ以上が定義可能**
- **さらにフォーマットを定義することでメッセージやコマンド定義の拡張が可能**
- **各種モジュールの接続は基本的にI²Cバス接続**
- **モジュール単体のシリアル接続も可能**
- **物理的に問題ないなら最大100モジュール以上接続可能**
- **プロトコルとサンプルプログラムを公開し、別途定義するライセンスの元での利用は無料**
- **モジュールを自作することで独自制御システムの構築も可能**

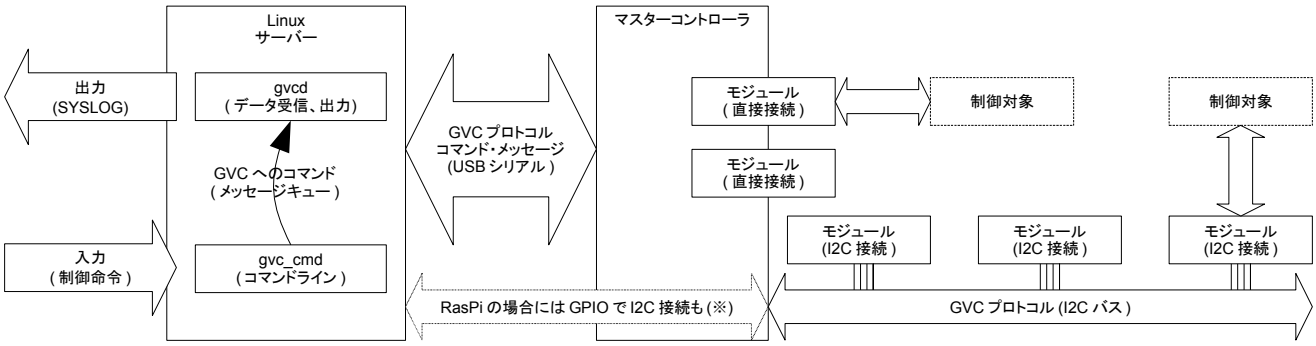
GVCのI²Cバスラインからの供給電圧は5Vを想定しています。USB給電でもある程度の動作は可能かと思いますが、大電流を必要とする場合には別途外部から供給する必要があります。また、モジュールに搭載するセンサーなどによっては3.3V供給が必要な場合は、個別のモジュールで供給電圧やI²Cのレベル変換をしてください。

GVCのハードウェアについては、別冊の『GVCハードウェアハンドブック』を参照してください。

2.0 構成

GVCの構成図を以下に示します。

図2-1 GVC構成図



※I²Cの信号電圧に注意(RasPiは3.3V、GVCは5V)

GVCは現在、Linuxサーバー上で動作するデーモン(gvcd)と、コマンドラインから各種制御を命令するgvc_cmd、そして各モジュールとI²Cバスで接続したマスターコントローラーをLinuxサーバーとUSBシリアル接続したのが標準的な構成となります。

マスターコントローラーは起動時にI²Cバスで接続されているモジュールをスキャンして、接続の有無を確認します。その後、特にLinuxサーバー側から命令がない限りはデータ取得系モジュール(例えば温度や湿度を返すようなもの)からデータを取得してLinuxサーバー側にGVCフォーマットでデータを一定時間毎に返すような動作をするのが標準的になります。GVC使用者は、マスターコントローラーに接続したモジュールやシステムの目的により、GVC使用者がその動作を変更することが可能です。

デーモンは、起動時に引数として渡したシリアルポートからGVCフォーマットのデータがやってくると、それを解析してSYSLOGに出力します。GVC使用者は必要に応じてこのデータを取得して処理をしてください。また、デーモンはコマンドラインからの命令があれば、タイミングを合わせてマスターコントローラーにその命令を送信します。マスターコントローラーは受けた命令に従って対象となるモジュールに命令を出して制御します。

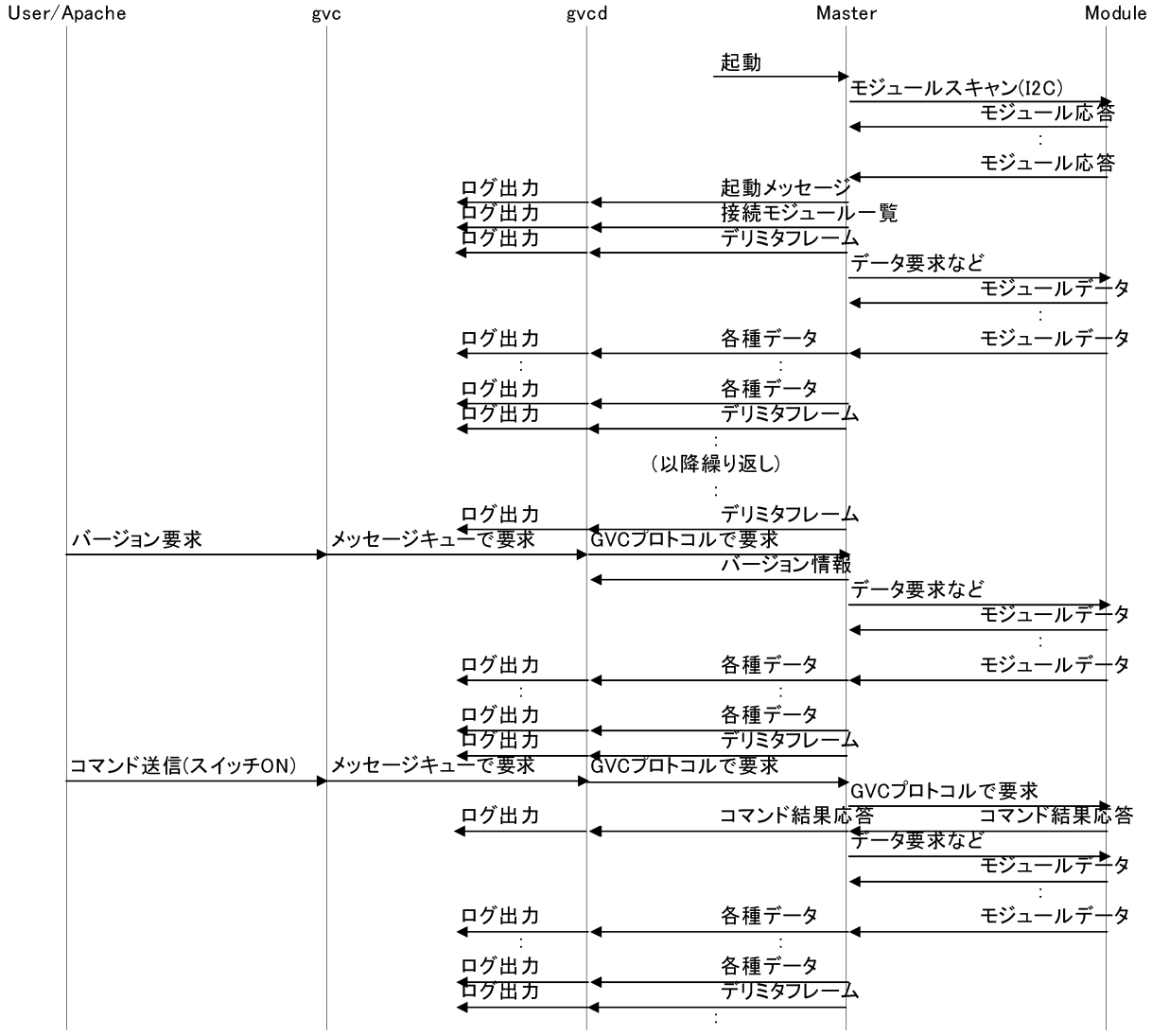
以上のように、GVCでは、プロトコルだけでもCLI-デーモン間、デーモン-マスターコントローラー間、そしてマスターコントローラー-モジュール間(I²Cバス接続の場合)が存在します。これらのプロトコルはコアとなる部分が全て統一されていますが、実際にはマスターコントローラーならびにモジュールで処理できるサイズに限界があったり、プログラムを作る際にはバイト境界調整を避けるために構造体を使用しないなどの注意が必要な場合があります。

2.1 プロトコルシーケンス

GVCで使用しているマスターコントローラーは給電されると自動的に起動します。すると、I²Cで接続されているモジュールをスキャンして接続モジュールの一覧を取得します。その後、起動メッセージをデーモンに送信すると同時に、必要に応じて接続モジュールの一覧などもデーモンに送り、一連の処理の区切りとしてデリミタフレームを送信します。

デーモンはコマンドラインインターフェースなどから命令を受け付けた場合には、デリミタフレームを待ってからマスターコントローラーに送信します。(※)　マスターコントローラーは受けたコマンドに基づいて、I²C接続されているモジュールに対してコマンドを送信します。モジュールでのコマンド処理などの結果は必要に応じてマスターコントローラーがデーモンに返して、デーモンがSYSLOGに出力します。

図2-2プロトコルシーケンスイメージ



※マスターコントローラー(Master)側がモジュールとやりとりをしているときにデーモン側からコマンドを送ると、マスターコントローラー側で取りこぼしが発生する可能性があるため、マスターコントローラー側がモジュールなどとやり取りをしていないデリミタフレームを送っている間にコマンドを送信する。

2.2 プロトコルフォーマット

GVCプロトコルには大きく分けて「メッセージ」と「コマンド」というものがあります。コマンドは文字通り命令で、マスターコントローラーやモジュールに対しての命令となります。メッセージは逆にマスターコントローラーやモジュールからの情報になります。ただし、便宜上コマンドはメッセージの一つに含まれます。

- **コマンドライン-デーモン間コマンド(メッセージキュー)**
- **デーモン-マスターコントローラー間メッセージ(シリアルUSB)**
- **マスターコントローラーモジュール間メッセージ(I²Cバス)**
- **デーモン-マスターコントローラー間コマンド(シリアルUSB)**
- **マスターコントローラーモジュール間コマンド(I²Cバス)**

※これらはやり取りの仕組みの制限などにより、サイズやタイプに違いが生じているので注意すること。

図2-3 コマンドライン - デーモン間コマンド (メッセージキュー)

0	1	2	3	4	5	6	…	n-1	n
gvc_num	msg_type	dev_num	format	cmd	data_len		not use		

※gvc_numは対象GVCの番号(とりあえず現状は0x01として一台のみが対象)
※msg_typeはコマンド(0x05)か、それ以外かを指定。(TBD)
※dev_numは対象GVCが保持しているモジュールデバイスの番号。1はGVC自身に対しての命令。
※formatは対象デバイス向けにあとに続くコマンドやデータの意味を示す。
※data_lenはこのあとに続くdataの長さとなる。データ無しの場合にはコマンドだけなので0。
※メッセージキューではchecksumは添付しない(テーブルアライメントの関係で)
※全体としてマスターコントローラーのBUFF_SIZEを越えないこと。

図2-4 デーモン - マスターコントローラー間メッセージ(シリアルUSB)

データありの場合の例(マスターコントローラーからの情報)

0	1	2	3	4	5	6	…	n-1	n
msg_type	dev_num	format	cmd	data_len		data			checksum

※dev_numは対象GVCが保持しているモジュールデバイスの番号。1はGVC自身に対しての命令。
※msg_typeはコマンド(0x05)か、それ以外かを指定。(TBD)
※formatは対象デバイス向けにあとに続くコマンドやデータの意味を示す。
※data_lenはこのあとに続くdataの長さとなる。データ無しの場合にはコマンドだけなので0。

図2-5 マスターコントローラーモジュール間コマンド(I²Cバス)

データありの場合の例

0	1	2	3	4	…	n-1	n
format	cmd	data_len		data			checksum

※formatは対象デバイス向けにあとに続くコマンドやデータの意味を示す。

上記のように、各伝送経路で使用するコマンドやメッセージは、上位経路でも内容を内包した状態になる。これにより、プログラムの流用性や、内部処理の統一性を持たせることが出来るようにしている。

これらの各種コマンドやメッセージ、そしてサンプルプログラムを次頁以降で解説します。

3.0 コマンドライン - デーモン間(メッセージキュー)コマンド

GVCではコマンドラインとデーモンとの間は、プロセス間通信としてメッセージキューを利用しています。デーモンそのものはシリアルポート(USBデバイス)などを握る必要があるためにroot権限で動作しますが、コマンドそのものはユーザー権限でも命令が送れるように、メッセージキューの属性を666にすることで、メッセージキューIDなどを合わせれば、GVC使用者が自分でこれから解説するコマンドを発行するプログラムを作ることも可能です。具体的には別項のサンプルプログラムを参照してください。

Linuxサーバーではメッセージキューの仕様上、メッセージそのもののサイズは固定です。ですので、何かデータを送りたい場合にはデータの長さをセットする必要があります。また一つのキューでは収まらない場合には分割して送信するなどの処理をしなければなりません(が、それについてはTBD)。

図3-1 コマンドライン-デーモン間(メッセージキュー)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	format	cmd	data_len		data		

全体長　　n bytes（※マスターコントローラーのBUFF_SIZEを越えないこと）

0　　対象マスターコントローラー番号
1　　メッセージタイプ(GVCへのコマンド、0x05)
2　　デバイスモジュールID(I²Cのアドレス)
3　　メッセージフォーマット（デバイス別に定義）
4　　コマンド
5-6　　データ長(dataの長さ)
7...n　　データ

※gvc_numは対象マスターコントローラーの番号(とりあえず現状は0x01として一台のみが対象)
※dev_numは対象マスターコントローラーが保持しているモジュールデバイスの番号。1はマスターコントローラー自身に対しての命令。
※formatは対象デバイス向けにあとに続くコマンドやデータの意味を示す。
※data_lenはこのあとに続くdataの長さとなる。データ無しの場合にはコマンドだけなので0。
※全体としてマスターコントローラーのBUFF_SIZEを越えないこと。

コマンド系は、デーモンの制御を対象としたものと、マスターコントローラーの制御を対象としたものとに大別できます。

マスターコントローラーの制御を対象としたコマンドがコマンドラインから命令される場合には、その命令をデーモンが受けると、先頭の対象マスターコントローラーIDを削除してからマスターコントローラーに送ります。各I/Fでのフォーマットの違いに注意してください。

次頁以降で、定義済みコマンドについて解説していきます。

3. 0x01マスターコントローラーバージョン情報要求

マスターコントローラーのバージョンを要求する。このコマンドを受け取ったデーモンは、マスターコントローラーに対してバージョン情報を要求する。結果、マスターコントローラーのバージョン情報を含むメッセージが帰ってきたらSYSLOGに出力する。

図3-0x01 マスターコントローラーバージョン情報要求(0x01)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	0x01	format	0x01	0x0000		----		

全体長　7 bytes（※キューサイズの残りの部分は無視）

0　対象マスターコントローラー番号
1　0x05（GVCへの命令）
2　0x01（マスターコントローラー自身への命令）
3　メッセージフォーマット
　0x01　通常テキストフォーマット
　0x10　カンマ区切りフォーマット
　0x20　TAB区切りフォーマット…とか(TBD)
4　0x01（バージョン情報要求）
5-6　0x0000（データはないので）
7...n　未使用

3. 0x02 マスターコントローラー接続モジュール情報要求

マスターコントローラーに接続されているモジュールの情報を要求する。このコマンドを受け取ったデーモンは、マスターコントローラーに対してモジュール一覧を要求する。結果、モジュール一覧情報を含むメッセージが帰ってきたらSYSLOGに出力する。

図3-0x02 接続モジュール情報要求(0x02)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	0x01	format	0x02	0x0000		----		

全体長　7 bytes（※キューサイズの残りの部分は無視）

0　対象マスターコントローラー番号
1　0x05（GVCへの命令）
2　0x01（マスターコントローラー自身への命令）
3　メッセージフォーマット
　0x01　通常テキストフォーマット
　0x10　カンマ区切りフォーマット
　0x20　TAB区切りフォーマット…とか(TBD)
4　0x02（マスターコントローラー接続モジュール情報要求）
5-6　0x0000（データはないので）
7...n　未使用

3. 0x03 モジュールデータ取得要求

マスターコントローラーに対して各種データを要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対して各種データを要求するコマンドを送ります。結果、データを含むメッセージが帰ってきたらSYSLOGに出力します。

図3-0x03 モジュールデータ取得要求(0x03)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	format	0x03	0x0000		----		

全体長　7 bytes（※キューサイズの残りの部分は無視）

0　対象マスターコントローラー番号
1　0x05（GVCへの命令）
2　対象デバイスモジュールID(I²Cのアドレス)
　0xFF　接続している全モジュールに対してデータ取得要求
3　メッセージフォーマット
　0x01　通常テキストフォーマット
　0x10　カンマ区切りフォーマット
　0x20　TAB区切りフォーマット…とか(TBD)
4　0x03（モジュールデータ取得要求）
5-6　0x0000（データはないので）
7...n　未使用

3.0x20 スイッチOFF要求

マスターコントローラーに対して接続されている指定モジュールのスイッチOFFを要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対して指定するモジュールのスイッチをOFFにするコマンドを送ります。結果を含むメッセージが帰ってきたらSYSLOGに出力します。

図3-0x20 スイッチOFF要求(0x20)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	----	0x20	0x0000		----		

全体長　7 bytes（※キューサイズの残りの部分は無視）

0　対象マスターコントローラー番号
1　0x05（GVCへの命令）
2　対象デバイスモジュールID(I²Cのアドレス)
　0xFF　接続している全モジュールに対してスイッチOFF要求
3　未使用
4　0x20（スイッチOFF要求）
5-6　0x0000（データはないので）
7...n　未使用

3.0x21 スイッチON要求

マスターコントローラーに対して接続されている指定モジュールのスイッチONを要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対して指定するモジュールのスイッチをONにするコマンドを送ります。結果を含むメッセージが帰ってきたらSYSLOGに出力します。

図3-0x21 スイッチOFF要求(0x21)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	----	0x21	0x0000		----		

全体長　7 bytes（※キューサイズの残りの部分は無視）

0　対象マスターコントローラー番号
1　0x05（GVCへの命令）
2　対象デバイスモジュールID(I²Cのアドレス)
　0xFF　接続している全モジュールに対してスイッチON要求
3　未使用
4　0x21（スイッチON要求）
5-6　0x0000（データはないので）
7...n　未使用

3.0x2f スイッチ状態要求

マスターコントローラーに対して接続されている指定モジュールのスイッチ状態を要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対して指定するモジュールのスイッチの状態を要求するコマンドを送ります。結果を含むメッセージが帰ってきたらSYSLOGに出力します。

図3-0x2f スイッチOFF要求(0x2f)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	----	0x2f	0x0000		----		

全体長　　7 bytes（※キューサイズの残りの部分は無視）

0　　対象マスターコントローラー番号
1　　0x05（GVCへの命令）
2　　対象デバイスモジュールID(I²Cのアドレス)
3　　未使用
4　　0x2f（スイッチ状態要求）
5-6　　0x0000（データはないので）
7...n　　未使用

3.0x7e マスターコントローラーリスタート/リセット要求(TBD)

マスターコントローラーのリスタート/リセットを要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対してリスタート/リセットを要求します。結果、マスターコントローラーからのデータをSYSLOGに出力します。

図3-0x7e マスターコントローラーリスタート/リセット情報要求(0x7e)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	0x01	----	0x7e	0x0000		----		

全体長　7 bytes（※キューサイズの残りの部分は無視）

0　対象マスターコントローラー番号
1　0x05（GVCへの命令）
2　0x01（マスターコントローラー自身への命令）
3　未使用
4　0x7e（マスターコントローラーリスタート/リセット要求）
5-6　0x0000（データはないので）
7...n　未使用

3.0x7f マスターコントローラー停止要求

マスターコントローラーの動作停止を要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対して停止を要求します。結果、停止メッセージが来たらSYSLOGに出力します。

図3-0x7f マスターコントローラー停止要求(0x7f)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	0x01	----	0x7f	0x0000		----		

全体長　7 bytes（※キューサイズの残りの部分は無視）

0　対象マスターコントローラー番号
1　0x05（GVCへの命令）
2　0x01（マスターコントローラー自身への命令）
3　未使用
4　0x7f（マスターコントローラー停止要求）
5-6　0x0000（データはないので）
7...n　未使用

3.0x81 LCDデータ出力要求

マスターコントローラーに接続されているLCDに文字列データの表示などを要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対してデータを送ります。結果、マスターコントローラーからのデータをSYSLOGに出力します。

図3-0x81 LCDデータ出力要求(0x81)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	format	0x81	data_len		data		

全体長　7 ＋ data_len bytes（※キューサイズの残りの部分は無視）

0　対象マスターコントローラー番号
1　0x05（GVCへの命令）
2　対象デバイスモジュールID(I²Cのアドレス)
3　メッセージフォーマット
　0x01　通常テキストフォーマット
　0x??　LCDによって別途定義…とか(TBD)
4　0x81（LCDデータ出力要求）
5-6　データ長
7...n　データ

3.0x91 リモコンデータ送信要求

マスターコントローラーに接続されている赤外線リモコンモジュールにリモコンデータの送信を要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対してリモコンデータを送ります。結果、マスターコントローラーからのデータをSYSLOGに出力します。

図3-0x91 リモコンデータ送信要求(0x91)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	----	0x91	0x0000		----		

全体長　　7 bytes（※キューサイズの残りの部分は無視）

0　　対象マスターコントローラー番号
1　　0x05（GVCへの命令）
2　　対象デバイスモジュールID(I²Cのアドレス)
3　　未使用
4　　0x91（リモコンデータ送信要求）
5-6　　0x0000（データはないので）
7...n　　未使用

※ 4.0 デーモン - マスターコントローラー間(シリアル)メッセージの4.0x49 GVC_MSG_IRを参照
※ 5.0 マスターコントローラーモジュール間(I²C)コマンド&メッセージのリモコンデータ関連を参照

3.0x92 リモコンデータ受信要求

マスターコントローラーに接続されている赤外線リモコンモジュールにリモコン信号の受信を要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対してリモコンデータ受信要求を送ります。結果、マスターコントローラーからのデータをSYSLOGに出力します。

実際にリモコンデータが読み取れたかどうかは、別途**モジュールデータ取得要求**(0x03)で確認をすること。

図3-0x92 リモコンデータ受信要求(0x92)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	----	0x92	0x0000		----		

全体長　　7 bytes（※キューサイズの残りの部分は無視）

0　　対象マスターコントローラー番号
1　　0x05（GVCへの命令）
2　　対象デバイスモジュールID(I²Cのアドレス)
3　　未使用
4　　0x92（リモコンデータ受信要求）
5-6　　0x0000（データはないので）
7...n　　未使用

※ 4.0 デーモン - マスターコントローラー間(シリアル)メッセージの4.0x49 GVC_MSG_IRを参照
※ 5.0 マスターコントローラーモジュール間(I²C)コマンド&メッセージのリモコンデータ関連を参照

3.0x93 リモコンデータメモリ設定要求

マスターコントローラーに接続されている赤外線リモコンモジュールにリモコンデータの設定を要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対してリモコンデータを送ります。結果、マスターコントローラーからのデータをSYSLOGに出力します。

図3-0x93 リモコンデータメモリ設定要求(0x93)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	format	0x93	data_len		data		

全体長　7 ＋ data_len bytes（※キューサイズの残りの部分は無視）

0　対象マスターコントローラー番号
1　0x05（GVCへの命令）
2　対象デバイスモジュールID(I²Cのアドレス)
3　メッセージフォーマット(※)
　0x01　GVC標準ASCIIフォーマット(信号が00～FFのASCII文字に変換してある)
　0x10　GVC標準BYNALYフォーマット(信号が0x00～0xFFのバイナリデータのまま)
4　0x93（リモコンデータメモリ設定要求）
5-6　データ長
7...n　データ

※ 4.0 デーモン - マスターコントローラー間(シリアル)メッセージの4.0x49 GVC_MSG_IRを参照
※ 5.0 マスターコントローラーモジュール間(I²C)コマンド＆メッセージのリモコンデータ関連を参照

3.0x94 リモコンデータメモリ読出要求

マスターコントローラーに接続されている赤外線リモコンモジュールにリモコンデータを要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対して指定したメモリバンクの読み出し要求を送ります。結果、マスターコントローラーからのデータをSYSLOGに出力します。またファイルに保存する場合には、dataの中にファイル名がフルパスで入ります。

図3-0x94 リモコンデータメモリ読出要求(0x94)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	format	0x94	data_len		data		

全体長　7 ＋ data_len bytes（※キューサイズの残りの部分は無視）

0　対象マスターコントローラー番号
1　0x05（GVCへの命令）
2　対象デバイスモジュールID(I²Cのアドレス)
3　メッセージフォーマット(※)
　0x01　GVC標準ASCIIフォーマット(信号が00～FFのASCII文字に変換してある)
　0x10　GVC標準BYNALYフォーマット(信号が0x00～0xFFのバイナリデータのまま)
4　0x94（リモコンデータメモリ読出要求）
5-6　データ長
7...n　データ(デーモンにデータをファイル出力させる場合にファイル名が入る)

※ 4.0 デーモン - マスターコントローラー間(シリアル)メッセージの4.0x49 GVC_MSG_IRを参照
※ 5.0 マスターコントローラーモジュール間(I²C)コマンド＆メッセージのリモコンデータ関連を参照

3.0x95 リモコンデータメモリ削除要求

マスターコントローラーに接続されている赤外線リモコンモジュールにリモコンデータの削除を要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対して指定したメモリバンクの削除要求を送ります。結果、マスターコントローラーからのデータをSYSLOGに出力します。

図5-0x95 リモコンデータメモリ削除要求(0x95)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	----	0x95	data_len		data		

全体長　　7 bytes（※キューサイズの残りの部分は無視）

0　　対象マスターコントローラー番号
1　　0x05（GVCへの命令）
2　　対象デバイスモジュールID(I²Cのアドレス)
3　　未使用
4　　0x95（リモコンデータメモリ削除要求）
5-6　　データ長(データ長が0x0000の場合には、データバンク0のデータを削除)
7...n　　データ(データバンクを指定する場合)

※ 4.0 デーモン - マスターコントローラー間(シリアル)メッセージの4.0x49 GVC_MSG_IRを参照
※ 5.0 マスターコントローラーモジュール間(I²C)コマンド&メッセージのリモコンデータ関連を参照

3.0xa1 音声合成データ出力要求

マスターコントローラーに接続されている音声合成モジュールに音声データの送信を要求します。このコマンドを受け取ったデーモンは、マスターコントローラーに対して音声データを送ります。

図3-0xa1 音声合成データ出力要求(0xa1)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	dev_num	format	0xa1	data_len		data		

全体長　　7 ＋ data_len bytes（※キューサイズの残りの部分は無視）

0　　対象マスターコントローラー番号
1　　0x05（GVCへの命令）
2　　対象デバイスモジュールID(I²Cのアドレス)
3　　メッセージフォーマット
　　0x01　GVC標準ASCIIフォーマット(バイナリが00～FFのASCII文字に変換してある)
　　0x10　GVC標準BYNALYフォーマット(0x00～0xFFのバイナリデータのまま)
4　　0xa1（音声合成データ出力要求）
5-6　　データ長
7...n　　データ

3.0xff デーモン停止要求

デーモン(gvcd)の動作停止を要求します。このコマンドを受け取ったデーモンは、速やかに動作を停止して結果をSYSLOGに出力します。

図3-0xff デーモン停止要求(0xff)コマンドフォーマット

0	1	2	3	4	5	6	7	…	n
gvc_num	0x05	----	----	0xff	0x0000		----		

全体長　　7 bytes（※キューサイズの残りの部分は無視）

0　　0x00（GVCデーモンへの命令）
1　　0x05（GVCへの命令）
2　　未使用
3　　未使用
4　　0xff（GVCデーモン停止要求）
5-6　　0x0000（データはないので）
7...n　　未使用

4.0 デーモン - マスターコントローラー間(シリアル)メッセージ

デーモンとマスターコントローラーの間はUARTによるいわゆるシリアル通信を利用します。通信速度はユーザーが任意に設定できますが、標準的には9600N81(9600bps、ノンパリティ、8ビット、ストップビット1)としています。デーモンそのものはデバイスなどを握る必要があるためにroot権限で動作します。

GVCで使われているマスターコントローラーにはフロー制御はありませんので、メッセージの受信やコマンドの送信のタイミングには一定の決まり(ハンドシェイク)でやり取りする必要があります。デーモンは、通常時はマスターコントローラー側からのメッセージを受けつつ、コマンドラインからメッセージキューで命令を待っています。実際に命令があってコマンドを送信する場合には、マスターコントローラー側からデリミタメッセージを待ってコマンドを送信します。

マスターコントローラー側からはデリミタメッセージが最低3フレームが送られてきますので、最初のデリミタメッセージを検知した時点でマスターコントローラー側にコマンドの送信を開始して、マスターコントローラー側が取りこぼしなく(うまく)コマンドを受け付けるようにしてください。具体的には別項のサンプルプログラムを参照してください。

このように、9600bpsという高速(笑)通信ではありますが、メッセージやコマンドのやり取りの時間を少しでも短縮するために、デーモン□マスターコントローラー間通信ではデータ長を可変長として処理時間の削減を図っています。

図4-1 デーモン - マスターコントローラー間(シリアル)メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
msg_type	dev_num	format	cmd	data_len		data			checksum

全体長　　n bytes（※マスターコントローラーのBUFF_SIZEを越えないこと）

0　　メッセージタイプ
1　　デバイスモジュールID(I²Cのアドレス)
2　　メッセージフォーマット（デバイス別に定義）
3　　コマンド
4-5　　データ長(dataの長さ)
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※dev_numは対象マスターコントローラーが保持しているモジュールデバイスの番号。1はマスターコントローラー自身に対しての命令。
※formatは対象デバイス向けにあとに続くコマンドやデータの意味を示す。
※data_lenはこのあとに続くdataの長さとなる。データ無しの場合にはコマンドだけなので0。
※全体としてマスターコントローラーのBUFF_SIZEを越えないこと。

次頁以降で、メッセージタイプ別に解説します。

4. 0x01 (TBD)

4. 0x02 (TBD)

4. 0x03 (TBD)

4. 0x04 (TBD)

4.0x05 GVC_MSG_ENQ : マスターコントローラーへコマンド送信

デーモン(gvcd)がマスターコントローラーに対して各種コマンドを送信するメッセージ。ただしデーモンが動作する環境によってはバイト境界の自動補正がかかる場合があるので、プログラム上でテーブル定義をする場合には注意が必要です。

実際にはコマンドラインーデーモン間コマンドで受けたものから対象マスターコントローラー番号を除いたものを送ります。(参考: **3.0 コマンドライン-デーモン間(メッセージキュー)コマンド**)

このメッセージを受け取ったマスターコントローラーは、必要な処理をして、結果を情報メッセージ(タイプ：0x21)としてデーモンに返します。

図4-0x05 マスターコントローラーへコマンド送信(0x05)メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x05	dev_num	format	cmd	data_len		data			checksum

全体長　　6 ＋ data_len + 1 bytes（※マスターコントローラーのBUFF_SIZEを越えないこと）

0　　0x05（GVCへの命令）
1　　対象デバイスモジュールID(I²Cのアドレス ※1)
　　0x01　マスターコントローラー自身に対しての命令。
2　　メッセージフォーマット
　　0x01　GVC標準ASCIIフォーマット(バイナリが00～FFのASCII文字に変換してある)
　　0x10　GVC標準BYNALYフォーマット(0x00～0xFFのバイナリデータのまま)
　　0x??　その他使い勝手により別途定義…とか(TBD)
3　　コマンド(※2)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 REQUEST MODULE LIST(0x02)でモジュール一覧を取得してから別途モジュールに命令を出すこと
※2 **3.0 コマンドライン-デーモン間(メッセージキュー)コマンド**を参照すること

4. 0x06 (TBD)

4. 0x07 (TBD)

4. 0x08 (TBD)

4. 0x09 (TBD)

4. 0x0a (TBD)

4. 0x0b (TBD)

4. 0x0c (TBD)

4. 0x0d (TBD)

4. 0x0e (TBD)

4. 0x0f (TBD)

4.0x10 GVC_MSG_PICDO : MODULE DATA OUT ONLY (TBD)

マスターコントローラーに接続されているデータ取得モジュール(モジュールから見てデータをOutputする方向)からのデータが含まれるメッセージ。まだメッセージタイプを確定していない試験用モジュールなどで利用します。

フォーマットやデータは開発者が任意に指定可能です。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをそのままSYSLOGに出力します。

図4-0x10 PIC(12F1822) DATA ONLYメッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x10	dev_num	format	cmd	data_len		data			checksum

全体長　　6 ＋ data_len + 1 bytes（※マスターコントローラーのBUFF_SIZEを越えないこと）

0　　0x10（対象デバイスモジュールへの出力）
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　0x01　GVC標準ASCIIフォーマット(バイナリが00～FFのASCII文字に変換してある)
　0x03　dataは文字列で小数点表記の実数16文字(1.02345、0.0098765)
　0x10　GVC標準BYNALYフォーマット(0x00～0xFFのバイナリデータのまま)
　0x11　dataはshort intで2バイト
　0x12　dataはlong intで4バイト
　0x13　dataはfloat(IEEE754に則した単精度の浮動小数点)で4バイト
　0x??　その他使い勝手により別途定義…とか(TBD)
3　　コマンド(モジュールの出力方法など…TBD ※1)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 formatやcmd、dataは皆さんで自由に定義して、モジュールの仕様が固まったら、ぜひともGVCプロトコルにご連絡ください。仕様として反映します。

4. 0x11 (TBD)

4. 0x12 (TBD)

4. 0x13 (TBD)

4. 0x14 (TBD)

4. 0x15 (TBD)

4. 0x16 (TBD)

4. 0x17 (TBD)

4. 0x18 (TBD)

4. 0x19 (TBD)

4. 0x1a (TBD)

4. 0x1b (TBD)

4. 0x1c (TBD)

4. 0x1d (TBD)

4. 0x1e (TBD)

4. 0x1f (TBD)

4. 0x20 (TBD)

4.0x21 GVC_MSG_OTHER : START, RESET, INFO, etc

マスターコントローラーそのものからの各種情報メッセージ。起動時、リセット/リスタート時、また接続モジュール情報要求などの各種コマンドに対する応答などがこのメッセージタイプに含まれます。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x21 START, RESET, INFO, etcメッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x21	dev_num	format	cmd	data_len		data			checksum

全体長　　6 ＋ data_len + 1 bytes（※マスターコントローラーのBUFF_SIZEを越えないこと）

0　　0x21（対象デバイスモジュールへの出力）
1　　0x01（マスターコントローラー自身からの情報）
2　　メッセージフォーマット
　　0x01　通常テキストフォーマット
　　0x10　カンマ区切りフォーマット
　　0x20　TAB区切りフォーマット…とか(TBD)
3　　結果(情報種別…TBD)
　　0x01　初期情報、システム情報
　　0x11　通常情報
　　0x21　コマンド応答
　　0x81　エラー情報
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

4. 0x22 (TBD)
4. 0x23 (TBD)
4. 0x24 (TBD)
4. 0x25 (TBD)
4. 0x26 (TBD)
4. 0x27 (TBD)
4. 0x28 (TBD)
4. 0x29 (TBD)
4. 0x2a (TBD)
4. 0x2b (TBD)
4. 0x2c (TBD)
4. 0x2d (TBD)
4. 0x2e (TBD)
4. 0x2f (TBD)
4. 0x30 (TBD)
4. 0x31 (TBD)
4. 0x32 (TBD)
4. 0x33 (TBD)
4. 0x34 (TBD)
4. 0x35 (TBD)
4. 0x36 (TBD)
4. 0x37 (TBD)
4. 0x38 (TBD)
4. 0x39 (TBD)
4. 0x3a (TBD)
4. 0x3b (TBD)
4. 0x3c (TBD)
4. 0x3d (TBD)
4. 0x3e (TBD)
4. 0x3f (TBD)
4. 0x40 (TBD)

4.0x41 GVC_MSG_ACCE : 'A' Acceleration (TBD) 加速度

マスターコントローラーに接続されている加速度センサーモジュールからのデータが含まれるメッセージ。加速度センサーモジュールの仕様により、一軸分のデータの場合もあれば、三軸分のデータまで含まれる場合もあるし、またそのデータのフォーマットもいろいろです。

というよりも、GVCはリアルタイム処理を想定していませんので(シリアル通信の場合には速度が遅すぎるから)、加速度の値を取得する必要がある場合には、何が必要なのか、またその取得速度などについても吟味する必要があると思います。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x41 'A' Acceleration (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x41	dev_num	format	cmd	data_len		data			checksum

全体長　　6 ＋ data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　　0x41 (加速度情報)
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　　0x01　通常テキストフォーマット
　　0x02　カンマ区切りテキストフォーマット
　　0x03　TAB区切りテキストフォーマット
　　0x10　バイナリフォーマット
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　現在の加速度でX軸
　　0x02　現在の加速度でXとY軸が連続している
　　0x03　現在の加速度でXとYとZ軸が連続している
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。

4. 0x42 (TBD)

4.0x43 GVC_MSG_COMP: 'C' Compass (TBD) 方位

マスターコントローラーに接続されている方位センサーモジュールからのデータが含まれるメッセージ。方位センサーモジュールの仕様により、分解能などが違い、またそのデータのフォーマットもいろいろです。

また、GVCはリアルタイム処理を想定していませんので(シリアル通信の場合には速度が遅すぎるから)、今どの方向を向いているのかについて得るのは簡単ですが、どの方向からどの方向にどのような速度で向いたのかというようなリアルタイム的な利用をする場合には吟味する必要があると思います。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x43 'C' Compass (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x43	dev_num	format	cmd	data_len		data			checksum

全体長　　6 + data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　　0x43 (方位情報)
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　　0x01　通常テキストフォーマット(整数 0～360)
　　0x02　通常テキストフォーマット(整数 -180～180)
　　0x11　文字列で小数点表記の実数16文字(0～360)
　　0x12　文字列で小数点表記の実数16文字(-180～180)
　　0x21　short intのバイナリフォーマット(整数 0～360)
　　0x31　short intバイナリフォーマット(整数 -180～180)
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。

4.0x44 GVC_MSG_DIST: 'D' Distance (TBD) 距離

マスターコントローラーに接続されている距離センサーモジュールからのデータが含まれるメッセージ。距離センサーモジュールの仕様により、測定範囲や分解能などが違い、またそのデータのフォーマットもいろいろです。

また、GVCはリアルタイム処理を想定していませんので(シリアル通信の場合には速度が遅すぎるから)、現在の距離がどれくらいかについて得るのは簡単ですが、どの方向からどの方向にどのような速度で向いたのかというようなリアルタイム的な利用をする場合には吟味する必要があると思います。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x44 'D' Distance (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x44	dev_num	format	cmd	data_len		data			checksum

全体長　　6 ＋ data_len + 1 bytes（※マスターコントローラーのBUFF_SIZEを越えないこと）

0　　0x44（距離情報）
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　　0x01　現在の距離[m]、文字列で小数点表記の実数16文字
　　0x02　現在の距離[mm]、文字列で小数点表記の実数16文字
　　0x03　現在の距離[um]、文字列で小数点表記の実数16文字
　　0x04　現在の距離[nm]、文字列で小数点表記の実数16文字
　　0x05　現在の距離[km]、文字列で小数点表記の実数16文字
　　0x08　現在の距離[光年]、文字列で小数点表記の実数16文字
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。

4.0x45 GVC_MSG_EARTH: 'E' Earth (TBD) GPS、地図

マスターコントローラーに接続されているGPSモジュールからのデータが含まれるメッセージ。GPSモジュールの仕様により、緯度経度だけでなく、基本的にはGPSから得られる全てのデータを載せることも可能です。

ただしGVCはリアルタイム処理を想定していませんので(シリアル通信の場合には速度が遅すぎるから)、緯度経度以外のデータについては、何が必要なのか、またその取得速度などについても吟味する必要があると思います。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x45 'E' Earth (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x45	dev_num	format	cmd	data_len		data			checksum

全体長　　6 ＋ data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　　0x45 (GPS情報)
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD)
　　0x01　NEMA-0183フォーマット文字列
　　0x11　NEMA-0183から緯度経度の文字列、,区切り
　　0x12　NEMA-0183からGPSステータス/測位モードの文字列(※1)
　　0x13　NEMA-0183から測位時刻(UTC)の文字列
　　0x14　NEMA-0183から対地速度(ノット)の文字列
　　0x15　NEMA-0183から進行方向(真北=0度)の文字列
　　0x16　NEMA-0183から標高[m]の文字列
　　0x17　NEMA-0183からジオイド高[m]の文字列
　　0x18　NEMA-0183から測位モードの文字列
　　0x19　NEMA-0183から磁気偏差の文字列
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 0=受信不能、1=単独測位、2=DGPS、もしくはA=単独測位、D=DGPS、V=無効

参考：http://www.gpscompass.jp/info.html

4. 0x46 (TBD)

4.0x47 GVC_MSG_GYRO : 'G' Gyro (TBD) ジャイロ

マスターコントローラーに接続されている方位センサーモジュールからのデータが含まれるメッセージ。方位センサーモジュールの仕様により、分解能などが違い、またそのデータのフォーマットもいろいろです。

また、GVCはリアルタイム処理を想定していませんので(シリアル通信の場合には速度が遅すぎるから)、今どの方向を向いているのかについて得るのは簡単ですが、どの方向からどの方向にどのような速度で向いたのかというようなリアルタイム的な利用をする場合には吟味する必要があると思います。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x47 'G' Gyro (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x47	dev_num	format	cmd	data_len		data			checksum

全体長　　6 + data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　　0x47 (ジャイロ情報)
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　　0x01　現在の向き/角度の通常テキスト(整数0～360)
　　0x02　現在の向き/角度の通常テキスト(整数-180～180)
　　0x11　現在の向き/角度の文字列で小数点表記の実数16文字(0～360)
　　0x12　現在の向き/角度の文字列で小数点表記の実数16文字(-180～180)
　　0x01　現在の向き/角度(Short intで整数0～360)
　　0x02　現在の向き/角度(Short intで整数-180～180)
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。

4.0x48 GVC_MSG_HUMI : 'H' Humidity (TBD) 湿度

マスターコントローラーに接続されている湿度センサーモジュールからのデータが含まれるメッセージ。湿度には相対湿度と絶対湿度があり、湿度センサーモジュールの仕様により、測定対象などが違い、またそのデータのフォーマットもいろいろです。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x48 'H' Humidity (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x48	dev_num	format	cmd	data_len		data			checksum

全体長　　6 + data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　　0x48 (湿度情報)
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　　0x01　現在の相対湿度(Short intで湿度を100倍した整数値)
　　0x11　現在の相対湿度[%]、文字列で小数点表記の実数16文字
　　0x21　現在の容積絶対湿度[g/㎥]、文字列で小数点表記の実数16文字
　　0x31　現在の重量絶対湿度[kg/kg(DA)]、文字列で小数点表記の実数16文字
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。

4.0x49 GVC_MSG_IR : 'I' Infrared (TBD) 赤外線(焦電)

マスターコントローラーに接続されている赤外線もしくは焦電センサーモジュールからのデータが含まれるメッセージ。赤外線リモコンなどのデータや、物体が出す赤外線を焦電センサーが検知するが、焦電センサーモジュールの仕様により、データのフォーマットもいろいろです。

また、GVCはリアルタイム処理を想定していませんので(シリアル通信の場合には速度が遅すぎるから)、焦電センサーモジュールがデータを送る時の状況について得るのは簡単ですが、リモコンデータのようなデータ量が多い場合や、データを送るシーケンスの間に物体を検知したかどうか、というような利用をする場合には、モジュール側の作りこみを吟味する必要があると思います。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x49 'I' Infrared (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x41	dev_num	format	cmd	data_len		data			checksum

全体長　6 + data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　0x49 (赤外線情報)
1　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　メッセージフォーマット(TBD ※2)
- 0x01　現在の焦電センサーの状態(0x00:未検知、0x01:検知)
- 0x11　現在の焦電センサーの状態(Short intでPICのレジスタ値とか)
- 0x21　赤外線リモコンデータの通常テキスト(ビット格納タイプ ※1)
- 0x22　赤外線リモコンデータの通常テキスト(01表記タイプ ※1)
- 0x23　赤外線リモコンデータの通常テキストカンマ区切り(ビット格納タイプ ※1)
- 0x24　赤外線リモコンデータの通常テキストカンマ区切り(01表記タイプ ※1)
- 0x25　赤外線リモコンデータの通常テキストTAB区切り(ビット格納タイプ ※1)
- 0x26　赤外線リモコンデータの通常テキストTAB区切り(01表記タイプ ※1)
- 0x31　赤外線リモコンデータのバイナリデータ(※1)
- 0x81　モジュール情報(通常テキストフォーマット)
- 0x82　モジュール情報(カンマ区切りテキストフォーマット)
- 0x83　モジュール情報(TAB区切りテキストフォーマット)
- 0x8B　モジュール情報(バイナリフォーマット)

3　結果(情報種別…TBD)
- 0x01　正常取得
- 0x81　取得エラー(エラー情報はdataに)

4-5　データ長
6...n-1　データ
n　formatからデータの最後(n-1)までのチェックサム

※1 赤外線リモコンデータ(0x21)の場合は、dataのフォーマットは以下の通り(TBD)
・メモリバンク番号
・サンプリング間隔[us](バイナリの場合にはcharで0-255[us])
・H/Lデータ

※2 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。

4.0x4a GVC_MSG_JOY : 'J' Joystick (TBD) ジョイスティック、回転角

マスターコントローラーに接続されているジョイスティックモジュールからのデータが含まれるメッセージ。ジョイスティックモジュールの仕様により、送られてくるデータやその分解能などが違います。

また、GVCはリアルタイム処理を想定していませんので(シリアル通信の場合には速度が遅すぎるから)、反応速度を気にするような場合には、ジョイスティックそのものもモジュールも含め、どのような処理にするのか吟味する必要があると思います。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x4a 'J' Joystick (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x4a	dev_num	format	cmd	data_len		data			checksum

全体長　　6 + data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　　0x4a (ジョイスティック情報)
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD)
　　0x01　現在の状態
　　0x10　蓄積データ(※2)
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ(※1)
n　　formatからデータの最後(n-1)までのチェックサム

※1 dataに入るボタン押下状態のフォーマットは以下の通り(TBD)

　　0x11　上ボタン押下
　　0x12　下ボタン押下
　　0x13　右ボタン押下
　　0x14　左ボタン押下
　　0x15　Aボタン押下
　　0x16　Bボタン押下
　　0x17　L1ボタン押下
　　0x18　L2ボタン押下
　　0x1a　R1ボタン押下
　　0x1b　R2ボタン押下
　　0x21　Lアナログスティックの方向、量とか
　　0x22　Rアナログスティックの方向、量とか

※2 蓄積データの場合には連続した形でなんたらかんたら…(TBD)

4. 0x4b (TBD)

4.0x4c GVC_MSG_LIGHT : 'L' LIGHT (TBD) 照度

マスターコントローラーに接続されている照度センサーモジュールからのデータが含まれるメッセージ。照度センサーモジュールの仕様により、分解能などが違い、またそのデータのフォーマットもいろいろです。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x4c 'L' LIGHT (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x4c	dev_num	format	cmd	data_len		data			checksum

全体長　　6 ＋ data_len + 1 bytes（※マスターコントローラーのBUFF_SIZEを越えないこと）

0　　0x4c（照度情報）
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　　0x01　現在の照度(PIC直読みのポートの値)
　　0x11　現在の照度[lx(lm/㎡)]、文字列で小数点表記の実数16文字
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。

4.0x4d GVC_MSG_MAG : 'M' Magnetic (TBD) 磁力、磁気

マスターコントローラーに接続されている磁力/磁気センサーモジュールからのデータが含まれるメッセージ。磁力/磁気センサーモジュールの仕様により、検知検出のしかた、またその出力フォーマットもいろいろです。

また、GVCはリアルタイム処理を想定していませんので(シリアル通信の場合には速度が遅すぎるから)、磁力/磁気センサーモジュールがデータを送る時の状況について得るのは簡単ですが、データを送るシーケンスの間に磁力/磁気を検知したかどうか、というような利用をする場合には、モジュール側の作りこみを吟味する必要があると思います。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x4d 'M' Magnetic (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x4d	dev_num	format	cmd	data_len		data			checksum

全体長　6 + data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　0x4d (磁力、磁気情報)
1　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　メッセージフォーマット(TBD ※1)
　0x01　現在の磁気状態(ホールセンサ、charで0x00=L、0x01=Hとか)
　0x02　現在の磁気状態(磁気抵抗、Short intでPICのレジスタ値とか)
　0x03　現在の磁気状態(磁気センサー、charで0x00=OFF、0x01=ONとか)
　0x12　現在の磁気状態(磁気抵抗、文字列で小数点表記の実数16文字の電圧値とか)
　0x22　現在の磁気状態(磁気抵抗、文字列で小数点表記の実数16文字の抵抗値とか)
　0x81　モジュール情報(通常テキストフォーマット)
　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　0x83　モジュール情報(TAB区切りテキストフォーマット)
　0x8B　モジュール情報(バイナリフォーマット)
3　結果(情報種別…TBD)
　0x01　正常取得
　0x81　取得エラー(エラー情報はdataに)
4-5　データ長
6...n-1　データ
n　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。

4. 0x4e (TBD)

4. 0x4f (TBD)

4.0x50 GVC_MSG_PRES : 'P' Pressure (TBD) 圧力

マスターコントローラーに接続されている圧力センサーモジュールからのデータが含まれるメッセージ。圧力センサーモジュールの仕様により、測定範囲や分解能などが違い、またそのデータのフォーマットもいろいろです。

また、GVCはリアルタイム処理を想定していませんので(シリアル通信の場合には速度が遅すぎるから)、現在の圧力がどれくらいかについて得るのは簡単ですが、データを送るシーケンスの間に圧力を検知したかどうか、というような利用をする場合には、モジュール側の作りこみを吟味する必要があると思います。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x50 'P' Pressure (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x50	dev_num	format	cmd	data_len		data			checksum

全体長　　6 ＋ data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　　0x50 (圧力情報)
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　　0x01　現在の気圧[hPa]、文字列で小数点表記の実数16文字
　　0x02　現在の圧力状態(charで0x00=なし, 0x01=ありとか)
　　0x11　現在の圧力[mg/cm2]、文字列で小数点表記の実数16文字
　　0x12　現在の圧力[g/cm2]、文字列で小数点表記の実数16文字
　　0x13　現在の圧力[kg/cm2]、文字列で小数点表記の実数16文字
　　0x14　現在の圧力[t/cm2]、文字列で小数点表記の実数16文字
　　0x21　現在の圧力状態(Short intでPICのレジスタの値とか)
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。

4. 0x51 (TBD)

4.0x52 GVC_MSG_RAD : 'R' Radiation (TBD) 放射線

マスターコントローラーに接続されている放射線センサーモジュールからのデータが含まれるメッセージ。放射線センサーモジュールの仕様により、測れる放射線の種類や、測定範囲、分解能、単位などなどが違い、またそのデータのフォーマットもいろいろです。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x52 'R' Radiation (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x52	dev_num	format	cmd	data_len		data			checksum

全体長　　6 + data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　　0x52 (放射線情報)
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　　0x11　単位時間当たりのカウンタ値、文字列で小数点表記の実数16文字
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。
　　正直よくわからないのでどなたかよろしくお願いします

4.0x53 GVC_MSG_SIGHT : 'S' Sight (TBD) 視覚、色

マスターコントローラーに接続されている視覚/色センサーモジュールからのデータが含まれるメッセージ。視覚/色センサーモジュールの仕様により、視覚センサーの場合には画像データのフォーマット、色センサーの場合には測定色や、測定範囲、分解能、単位などなどいろいろと違います。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x53 'S' Sight (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x53	dev_num	format	cmd	data_len		data			checksum

全体長　6 + data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　0x53 (視覚、色情報)
1　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　メッセージフォーマット(TBD ※1)
- 0x01　RGB各色の値[LSB/lx]、文字列で小数点表記の実数16文字,区切りで、R,G,B,LSB
- 0x02　R色の値[LSB/lx]、文字列で小数点表記の実数16文字,区切りで、R,LSB
- 0x03　G色の値[LSB/lx]、文字列で小数点表記の実数16文字,区切りで、G,LSB
- 0x04　B色の値[LSB/lx]、文字列で小数点表記の実数16文字,区切りで、B,LSB
- 0x11　RGB各色の値[LSB/lx]、文字列で小数点表記の実数16文字,区切りで、R,G,B,Vref,分解能(512とか)
- 0x12　R色の値[LSB/lx]、文字列で小数点表記の実数16文字,区切りで、R,Vref,分解能(512とか)
- 0x13　G色の値[LSB/lx]、文字列で小数点表記の実数16文字,区切りで、G,Vref,分解能(512とか)
- 0x14　B色の値[LSB/lx]、文字列で小数点表記の実数16文字,区切りで、B,Vref,分解能(512とか)
- 0x81　モジュール情報(通常テキストフォーマット)
- 0x82　モジュール情報(カンマ区切りテキストフォーマット)
- 0x83　モジュール情報(TAB区切りテキストフォーマット)
- 0x8B　モジュール情報(バイナリフォーマット)

3　結果(情報種別…TBD)
- 0x01　正常取得
- 0x81　取得エラー(エラー情報はdataに)

4-5　データ長
6...n-1　データ
n　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。

4.0x54 GVC_MSG_TEMP : 'T' Temperature (TBD) 温度

マスターコントローラーに接続されている温度センサーモジュールからのデータが含まれるメッセージ。温度センサーモジュールの仕様により、測定範囲や分解能などが違い、またそのデータのフォーマットもいろいろです。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x54 'T' Temperature (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x54	dev_num	format	cmd	data_len		data			checksum

全体長　　6 + data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　　0x54 (温度情報)
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　　0x01　現在の温度[° C]、摂氏、文字列で小数点表記の実数16文字
　　0x02　現在の温度[° F]、華氏、文字列で小数点表記の実数16文字
　　0x03　現在の温度[K]、絶対温度、文字列で小数点表記の実数16文字
　　0x11　現在の温度[° C]、摂氏、文字列で小数点表記の実数16文字, 区切りで、T, Vref, 分解能(512とか)
　　0x12　現在の温度[° F]、華氏、文字列で小数点表記の実数16文字, 区切りで、T, Vref, 分解能(512とか)
　　0x13　現在の温度[K]、絶対温度、文字列で小数点表記の実数16文字, 区切りで、T, Vref, 分解能(512とか)
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。

4.0x55 GVC_MSG_UV : 'U' UV (TBD) 紫外線

マスターコントローラーに接続されている紫外線センサーモジュールからのデータが含まれるメッセージ。紫外線センサーモジュールの仕様により、測れる紫外線の測定範囲、分解能、単位などなどが違い、またそのデータのフォーマットもいろいろです。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x55 'U' UV (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x55	dev_num	format	cmd	data_len		data			checksum

全体長　　6 + data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　　0x55 (紫外線情報)
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　　0x11　単位時間当たりの…？値、文字列で小数点表記の実数16文字
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条やり取りすること。
　　正直よくわからないのでどなたかよろしくお願いします

4. 0x56 (TBD)

4.0x57 GVC_MSG_IR : 'W' Wave (TBD) 音波

マスターコントローラーに接続されている音センサーモジュールからのデータが含まれるメッセージ。音センサーモジュールの仕様により、音の有無、音のレベル、音のデータ(サンプリングデータ？)など、範囲、分解能、単位、フォーマットなどなどがいろいろです。

デーモンはこのメッセージを受けると、マスターコントローラーの番号やデバイス番号、メッセージ種別ともにデータをSYSLOGに出力します。

図4-0x57 'W' Wave (TBD) メッセージフォーマット

0	1	2	3	4	5	6	…	n-1	n
0x57	dev_num	format	cmd	data_len		data			checksum

全体長　　6 ＋ data_len + 1 bytes (※マスターコントローラーのBUFF_SIZEを越えないこと)

0　　0x57 (音波情報)
1　　対象デバイスモジュールID(I²Cのアドレス、I²C接続でない場合には別途定義←TBD)
2　　メッセージフォーマット(TBD ※1)
　　0x01　？？？値
　　0x11　？？？値
　　0x21　？？？値
　　0x81　モジュール情報(通常テキストフォーマット)
　　0x82　モジュール情報(カンマ区切りテキストフォーマット)
　　0x83　モジュール情報(TAB区切りテキストフォーマット)
　　0x8B　モジュール情報(バイナリフォーマット)
3　　結果(情報種別…TBD)
　　0x01　正常取得
　　0x81　取得エラー(エラー情報はdataに)
4-5　　データ長
6...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※1 PICの値が直接来る場合には、基準電圧や想定センサーなどの諸条件もやり取りすること。
　　正直よくわからないのでどなたかよろしくお願いします

4. 0x58 (TBD)

4. 0x59 (TBD)

4. 0x5a (TBD)

4. 0x5b (TBD)

4. 0x5c (TBD)

4. 0x5d (TBD)

4. 0x5e (TBD)

4. 0x5f (TBD)

4. 0x60 (TBD)

4. 0x61 (TBD)

4. 0x62 (TBD)

4. 0x63 (TBD)

4. 0x64 (TBD)

4. 0x65 (TBD)

4. 0x66 (TBD)

4. 0x67 (TBD)

4. 0x68 (TBD)

4. 0x69 (TBD)

4. 0x6a (TBD)

4. 0x6b (TBD)

4. 0x6c (TBD)

4. 0x6d (TBD)

4. 0x6e (TBD)

4. 0x6f (TBD)

4. 0x70 (TBD)

4. 0x71 (TBD)

4. 0x72 (TBD)

4. 0x73 (TBD)

4. 0x74 (TBD)

4. 0x75 (TBD)

4. 0x76 (TBD)

4. 0x77 (TBD)

4. 0x78 (TBD)

4. 0x79 (TBD)

4. 0x7a (TBD)

4. 0x7b (TBD)

4. 0x7c (TBD)

4. 0x7d (TBD)

4. 0x7e (TBD)

4. 0x7f (TBD)

5.0 マスターコントローラーモジュール間(I²C)コマンド&メッセージ

マスターコントローラーと各種モジュールの間は、I²C(Inter-Integrated Circuit、I-squared-C＝アイ・スクエアド・シーが正式な呼称)という、同期クロック(SCL)と同期データ(SDA)の二本だけのいわゆるバス接続で通信します。SCLとSDAはそれぞれVddと抵抗で接続されプルアップされています。GVCではこの2本のほか、特段電力を必要とするモジュールがない限り、各モジュールへの電源供給とプルアップを兼ねたVddとGNDをあわせた合計4本でモジュールを接続します。

なおGVCのI²Cでは一斉同報には対応しません。またI²C自体が0x00から0x07と、0x78から0x7fまでの計16個のアドレスについては予約しているために、個別のモジュール製作者はこれらに注意してください。マスターコントローラーやPICはI²Cでやり取りをしている間は、たいていその他の割り込みを禁止していますので、処理時間にも注意してください。

別項でPIC用のI²Cをもちいるサンプルソースを掲載していますので、そちらを参照していただいて、便利なモジュールを製作してどんどん公開してください。(TBD)

図5-1 マスターコントローラーモジュール間(I²C)コマンド/メッセージフォーマット(データなし)

0	1	2	3	4
format	cmd	0x0000		checksum

全体長　5（※マスターコントローラーのBUFF_SIZEを越えないこと）

0　メッセージフォーマット（デバイス別に定義）
1　コマンド/結果
2-3　0x0000（データはないので）
n　formatからデータの最後(n-1)までのチェックサム

図5-2 マスターコントローラーモジュール間(I²C) コマンド/メッセージフォーマット(データあり)

0	1	2	3	4	…	n-1	n
format	cmd	data_len		data			checksum

全体長　n（※マスターコントローラーのBUFF_SIZEを越えないこと）

0　メッセージフォーマット（デバイス別に定義）
1　コマンド/結果
2-3　データ長(dataの長さ)
4...n-1　データ
n　formatからデータの最後(n-1)までのチェックサム

※formatは対象デバイス向けにあとに続くコマンドやデータの意味を示す。
※data_lenはこのあとに続くdataの長さとなる。データ無しの場合にはコマンドだけなので0。
※全体としてマスターコントローラーのBUFF_SIZEを越えないこと。

次頁以降で、メッセージタイプ別に解説します。

5. 0x03 モジュールデータ取得要求

モジュールに対して各種データを要求します。このコマンドを受け取ったモジュールは折り返し要求されたデータを返します。結果、データを受け取ったマスターコントローラーはデーモンに対してデータを送ります。

図5-0x03 モジュールデータ取得要求(0x03)コマンドフォーマット

0	1	2	3	4
format	0x03	0x0000		checksum

全体長　　4 + 1 bytes

0　　メッセージフォーマット（デバイス別に定義）
1　　**0x03**（モジュールデータ取得要求）
2-3　　**0x0000**（データはないので）
4　　formatからデータの最後(n-1)までのチェックサム

5.0x20 スイッチOFF要求

モジュールに対してスイッチOFFを要求します。このコマンドを受け取ったモジュールはスイッチをOFFにして結果を返します。

図5-0x20 スイッチOFF要求(0x20)コマンドフォーマット

0	1	2	3	4
format	0x20	0x0000		checksum

全体長　　4 + 1 bytes

0　　未使用(たとえばディマーのような動作が複数指定可能なら、dataで指定して処理すべき)
1　　0x20 (スイッチOFF要求)
2-3　　0x0000 (データはないので)
4　　format(not use)からデータの最後(n-1)までのチェックサム

5.0x21 スイッチON要求

モジュールに対してスイッチONを要求します。このコマンドを受け取ったモジュールはスイッチをONにして結果を返します。

図5-0x21 スイッチON要求(0x21)コマンドフォーマット

0	1	2	3	4
format	0x21	0x0000		checksum

全体長　　4 + 1 bytes

0　　未使用(たとえばディマーのような動作が複数指定可能なら、dataで指定して処理すべき)
1　　0x21 (スイッチON要求)
2-3　　0x0000 (データはないので)
4　　format(not use)からデータの最後(n-1)までのチェックサム

5.0x81 LCDデータ出力要求

モジュールに対してLCDデータ出力を要求します。このコマンドを受け取ったモジュールは受け取ったデータをLCDに出力して結果を返します。

図5-0x81 LCDデータ出力要求(0x81)コマンドフォーマット

0	1	2	3	4	…	n-1	n
format	0x81	data_len		data			checksum

全体長　　**4 ＋ data_len + 1 bytes**

0　　メッセージフォーマット
　0x01　通常テキストフォーマット
　0x??　LCDによって別途定義…とか(TBD)
1　　**0x81** (LCDデータ出力要求)
2-3　　データ長
4...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

5.0x91 リモコンデータ送信要求

モジュールに対してリモコンデータの送信を要求します。このコマンドを受け取ったモジュールは受け取ったデータをリモコン信号として送信して結果を返します。

図5-0x91 リモコンデータ送信要求(0x91)コマンドフォーマット

0	1	2	3	4
----	0x91	0x0000		checksum

全体長　　4 ＋ 1 bytes

0　　未使用
1　　0x91（リモコンデータ出力要求）
2-3　　0x0000（データはないので）
4...n-1　　未使用
n　　formatからデータの最後(n-1)までのチェックサム

※ 3.0 コマンドライン - デーモン間(メッセージキュー)コマンドの3.0x9? リモコンデータ関連を参照
※ 4.0 デーモン - マスターコントローラー間(シリアル)メッセージの4.0x49 GVC_MSG_IRを参照

5.0x92 リモコンデータ受信要求

モジュールに対してリモコンデータの受信を要求します。このコマンドを受け取ったモジュールはリモコンデータの受信状態となり、読み取りバンク番号を結果として返します。実際にリモコンデータが読み取れたかどうかは、別途**モジュールデータ取得要求**(0x03)で確認をすること。

図5-0x92 リモコンデータ受信要求(0x92)コマンドフォーマット

0	1	2	3	4
----	0x92	0x0000		checksum

全体長　　4 + 1 bytes

0　　未使用
1　　0x92（リモコンデータ受信要求）
2-3　　0x0000（データはないので）
4　　format(not use)からデータの最後(n-1)までのチェックサム

※ 3.0 コマンドライン - デーモン間(メッセージキュー)コマンドの3.0x9? リモコンデータ関連を参照
※ 4.0 デーモン - マスターコントローラー間(シリアル)メッセージの4.0x49 GVC_MSG_IRを参照

5.0x93 リモコンデータメモリ設定要求

モジュールに対してリモコンデータを内部メモリに設定する要求します。このコマンドを受け取ったモジュールは受け取ったデータを空いているメモリ領域に格納し、書き込んだメモリバンク番号などの結果を返します。**モジュールはリモコンデータをメモリに格納するだけで、送信は別途コマンドで要求する必要があります。**

図5-0x93 リモコンデータメモリ設定要求(0x93)コマンドフォーマット

0	1	2	3	4	…	n-1	n
format	0x93	data_len		data			checksum

全体長　　4 ＋ data_len + 1 bytes

0　　メッセージフォーマット(※)
　0x01　GVC標準ASCIIフォーマット(信号が00～FFのASCII文字に変換してある)
　0x10　GVC標準BYNALYフォーマット(信号が0x00～0xFFのバイナリデータのまま)
1　　0x93（リモコンデータメモリ設定要求）
2-3　　データ長
4...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

※ 3.0 コマンドライン - デーモン間(メッセージキュー)コマンドの3.0x9? リモコンデータ関連を参照
※ 4.0 デーモン - マスターコントローラー間(シリアル)メッセージの4.0x49 GVC_MSG_IRを参照

5.0x94 リモコンデータメモリ読出要求

モジュールに対して内部メモリに書き込み(保持)されているリモコンデータを要求します。このコマンドを受け取ったモジュールは指定されたメモリバングにあるデータを指定されたフォーマットで返します。

図5-0x94 リモコンデータメモリ読出要求(0x94)コマンドフォーマット

0	1	2	3	4	…	n-1	n
format	0x94	data_len		data			checksum

全体長　　4 ＋ data_len + 1bytes

0　　メッセージフォーマット(※1)
　0x01　GVC標準ASCIIフォーマット(信号が00～FFのASCII文字に変換してある)
　0x10　GVC標準BYNALYフォーマット(信号が0x00～0xFFのバイナリデータのまま)
　0x??　その他使い勝手により別途定義…とか(TBD)
1　　0x94（リモコンデータメモリ読出要求）
2-3　　データ長
4...n-1　　データ(メモリバンク番号1バイト)
n　　formatからデータの最後(n-1)までのチェックサム

※ 3.0 コマンドライン - デーモン間(メッセージキュー)コマンドの3.0x9? リモコンデータ関連を参照
※ 4.0 デーモン - マスターコントローラー間(シリアル)メッセージの4.0x49 GVC_MSG_IRを参照

5.0x95 リモコンデータメモリ削除要求

モジュールに対して内部メモリに書き込み(保持)されているリモコンデータの削除要求します。このコマンドを受け取ったモジュールは指定されたメモリバングにあるデータを削除します。

図5-0x95 リモコンデータメモリ削除要求(0x94)コマンドフォーマット

0	1	2	3	4
----	0x95	0x0000		checksum

全体長　　4 + 1 bytes

0　　未使用
1　　**0x95**（リモコンデータ受信要求）
2-3　　**0x0000**（データはないので）
4　　format(not use)からデータの最後(n-1)までのチェックサム

※ 3.0 コマンドライン - デーモン間(メッセージキュー)コマンドの3.0x9? リモコンデータ関連を参照
※ 4.0 デーモン - マスターコントローラー間(シリアル)メッセージの4.0x49 GVC_MSG_IRを参照

5.0xa1 音声合成データ出力要求

モジュールに対して音声合成データ出力を要求します。このコマンドを受け取ったモジュールは受け取ったデータを音声合成して出力して結果を返します。

図5-0xa1 音声合成データ出力要求(0xa1)コマンドフォーマット

0	1	2	3	4	…	n-1	n
format	0xa1	data_len		data			checksum

全体長　　4 ＋ data_len + 1bytes

0　　メッセージフォーマット
　0x01　GVC標準ASCIIフォーマット(バイナリが00～FFのASCII文字に変換してある)
　0x10　GVC標準BYNALYフォーマット(0x00～0xFFのバイナリデータのまま)
1　　0xa1（音声合成データ出力要求）
2-3　　データ長
4...n-1　　データ
n　　formatからデータの最後(n-1)までのチェックサム

…と、5.xなんですが、

I2Cにはメッセージとコマンド？がある。というか、相手がPICとか、もしくはMPL115A2とかの場合にはマスターコントローラー側というかマスター側から、スレーブ側であるデバイスに対して何らかのコマンドなりを送って、その結果としてデータをいただく＝これをメッセージと呼ぶならば、となる。

MPL115A2のようなどこかの誰かが作ったデバイスの場合にはそれに従わないといけないが、PIC＋センサーとかデバイスと言う場合にはここのところは自分で決めないといけない。でも決めろというとひっちゃかめっちゃかというかになるので、やはり指針としてある程度定義してあげるのがいいんじゃないかと思います。

自分メモです。:-)

これは書きかけです。

2013.02.28

開発を始めてからほぼ一年目にして、やっと赤外線リモコンまでできたので、一気にドキュメントというかプロトコルというかの見直しをしました。PICの変更によって、CRCの導入と、コマンド／メッセージの扱いが非常に楽(I/F毎の加工が少ない)にしてみました。

あとは実際のプログラムをこの新しいバージョンにあわせて、Rev.2を第一版としてリリースできたら、と思います。

2013.12.25

クリスマスに何やってんだという話はおいといて、いつの間にかGVC用の汎用基板も海外に発注しちゃったり、赤外線リモコンの波形でうんうん悩んでUSB接続するオシロまで買っちゃったり、まぁそのおかげでPIC内部の処理をきれいにしてうちの怪しいエアコンもピピッと動作させることが出来たり…とまぁ、足掛け三年もかかりましたけど、ひとまずまともに動くようになり、ソースも清書したり、このドキュメントも清書したりして、だいぶまとまってきました。

実際自分が使わないセンサーとかのデータフォーマットなんて、そっち方面に詳しい人に定義してもらった方が皆が幸せになるのでいいと思うんで、間違ってGVCに手を出してしまったそこのあなた、一緒にプロトコルとか作っていきませんか？でもって私にも楽しいシステムを教えてください。:-)

そんなこんなでまだまだつづきますー