AutoKey　JIGU-Board ソフトウェア仕様書

2009/1/31

yama

yama_e@users.sourceforge.jp

注意！

変更履歴

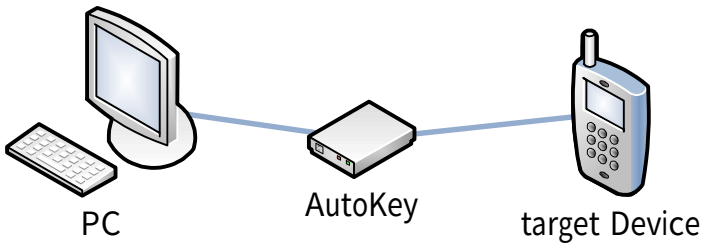
2009/1/31　新規作成 yama

目次

○プロジェクトの目的

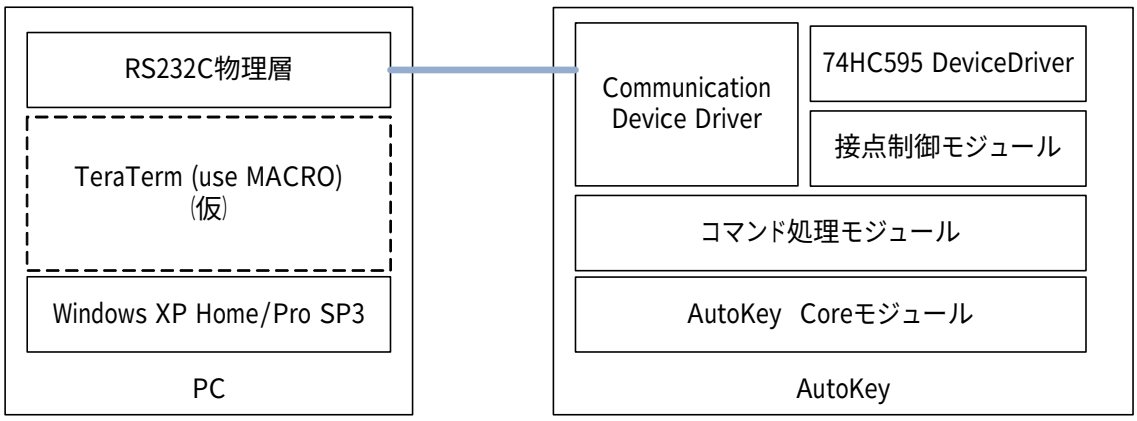
近年組込みシステムの品質向上大きな課題として取り上げられる機会も多くなり、ソフトウェア品質保証・ソフトウェアテストに関する書籍などを見かける機会が増えたものではないかと思います。
本プロジェクトは組込みシステムのソフトウェアテストフェーズにおいて一定の品質の確保と効率的なテスト作業を行うために役立つツールを提供します。特に単調なキー入力作業はテスターのモチベーションが下がるだけではなくそれによる作業ミス、注意不足からくるインシデントが操作ミスなのかフォールトなのかを判断できなくなる可能性があり繰り返し安定したテストを行うために本プロジェクトの成果物が大変役に立つと思われます。

○システム構成



Item	Description
PC	AutoKey へ接点信号のコマンド（指令）を送ります。 RS232C で接続します。
AutoKey	開発対象です。 PC からコマンドを受け接点信号を制御します
Target Device	制御対象です。 キーの接点を電気的に AutoKey により ON/OFF されます。

○ソフトウェア構成



Item	Description
Windows XP Home/Pro SP3	テスト環境の都合で Windows XP Home/Pro SP3 としていますが、それ以外でも使えると思います。
TeraTem	TeraTerm のマクロ機能を使用します。
RS232C 物理層	USB タイプの RS232C ポートでも対応可能です。 ただし TeraTerm で使用できれば OK です
AutoKey Core Module	AutoKey のコア部分です。 起動処理およびコマンド処理への導き処理を行います。
コマンド処理モジュール	PC から受け取るコマンド処理を行います エラー処理や各コマンドに対する処理を含めます
Communication Device Driver	RS232C の送受信を行うためのデバイスドライバーです
接点制御モジュール	74HC595*へ接点信号を出力する前処理などを行います
74HC595 Device Driver	74HC595 を制御します

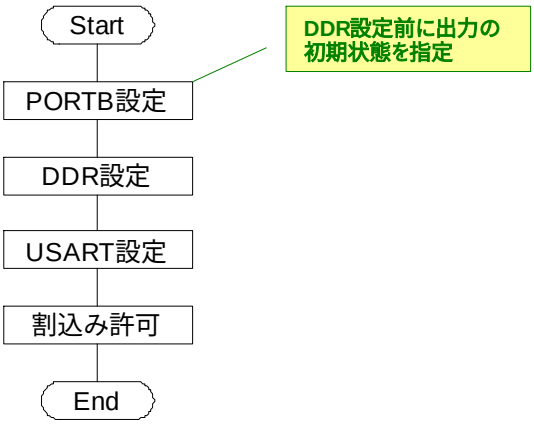
*シフトレジスターです。

○モジュール構成およびモジュール詳細

・AutoKey Core モジュール

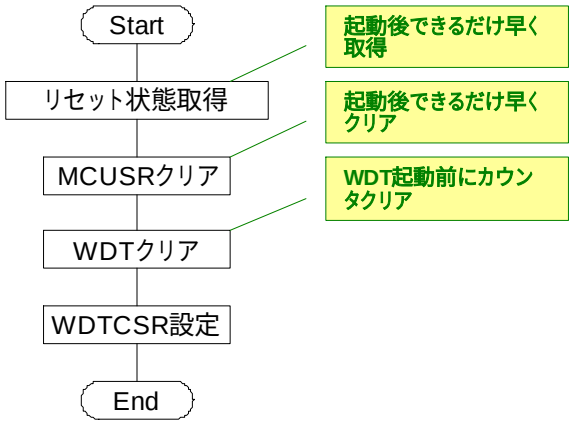
ファイル名	モジュール	内容
HardwareInit.c	HardwareInit()	ハードウェア初期化関数
	GetResetSetRegi()	起動時のリセット状態検査
AutoKey.c	Main()	起動処理

HardwareInit

項目	内容
モジュール名	HardwareInit
定義ファイル	HardwareInit.c
宣言ファイル	Module.h
書式	int hardwareInit(void)
概要	ハードウェアの初期化をします。 主に、IO の設定, USART の設定, 割り込み解除
引数	なし
戻り値	起動時ハードウェアエラー状態を戻します。 HARDINIT_NO_ERROR：ハードウェアエラーなし ※現バージョンはその他エラーなし
処理	

GetResetSetRegi

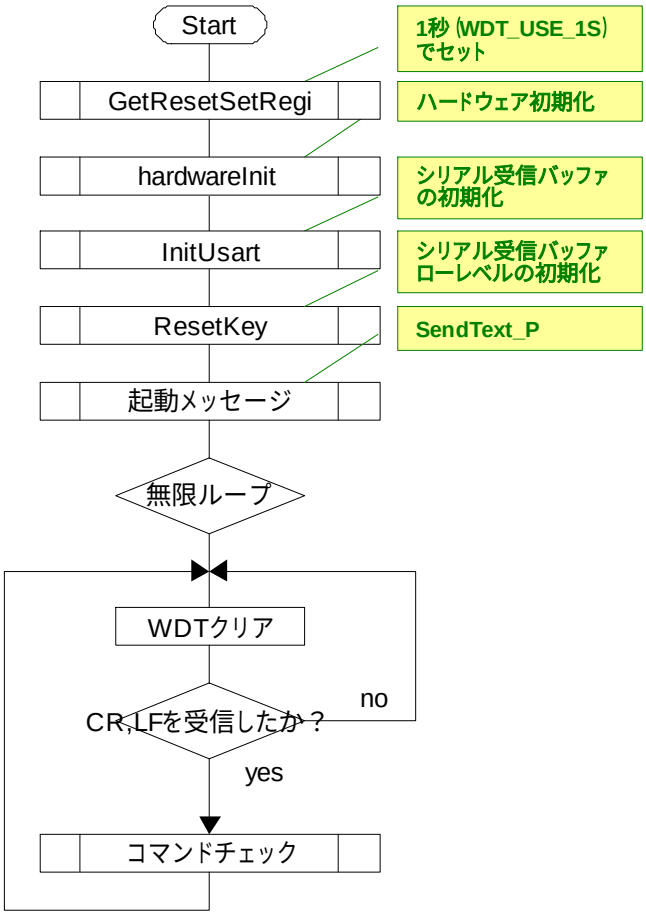
項目	内容
モジュール名	GetResetSetRegi
定義ファイル	HardwareInit.c

宣言ファイル	Module.h
書式	uint8_t GetResetSetRegi(uint8_t wdtcsr)
概要	ウォッチドッグの設定及び、起動時のリセット状態取得
引数	wdtcsr：ウォッチドッグリセット設定 WDT_NOUSE：ウォッチドッグを使用しない WDT_USE_1S：1 秒以上経過でウォッチドッグリセット
戻り値	リセット状態 PON_RESET：電源 ON リセット EXT_RESET：外部リセット BOR_RESET：電圧低下リセット WDT_RESET：ウォッチドッグタイマリセット
処理	

main

項目	内容
モジュール名	main
定義ファイル	AutoKey.c
宣言ファイル	---
書式	int main(void)
概要	int 型であるが return へ到達することはない。 起動後呼び出される
引数	なし
戻り値	0（但し、本関数内永久ループのため到達しない）

処理

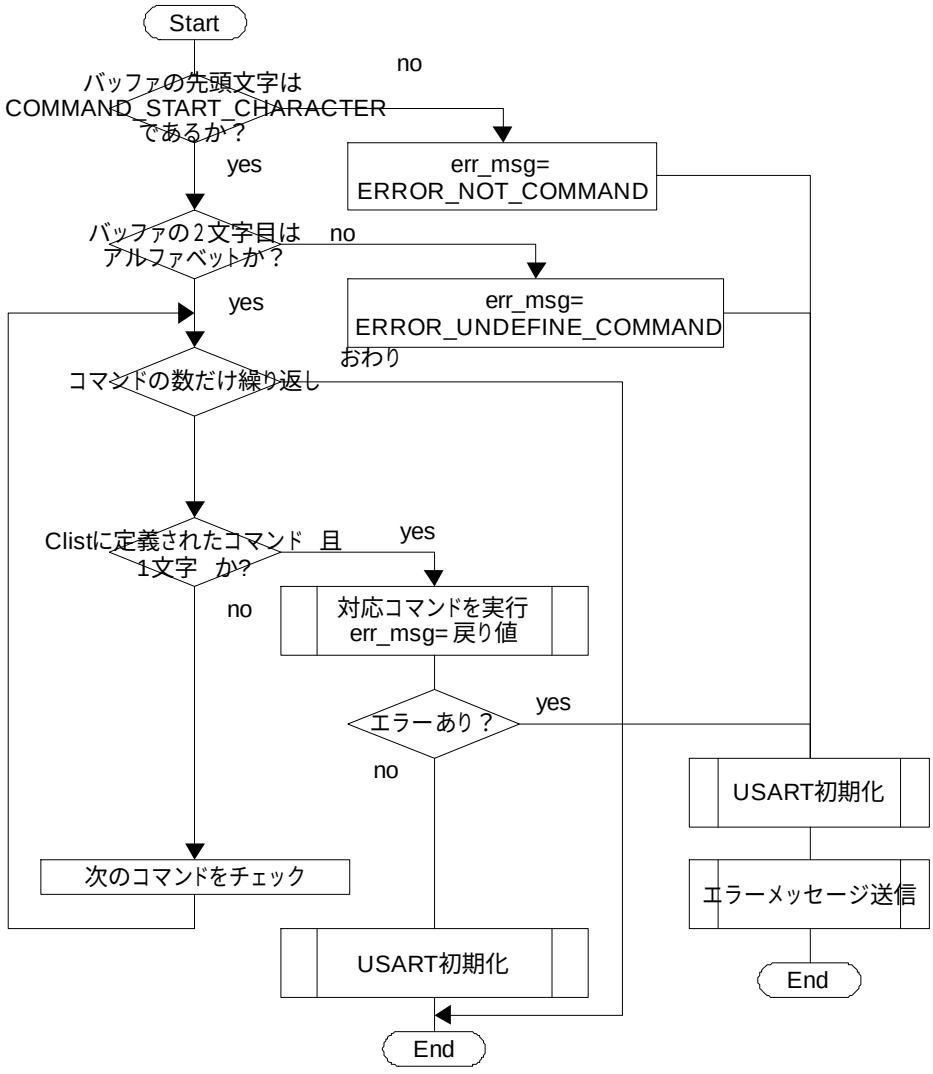


・コマンド処理モジュール

ファイル名	モジュール	内容
AutoKey.c	CommandCheck()	受信文字列をチェックしコマンドであれば実行する
	FuncKey()	接点出力コマンドの実体
	FuncVer()	バージョン確認コマンドの実体
	CommList CList[]	コマンド文字列とコマンド実体の関数ポインタの関連を定義した構造体の配列

CommandCheck

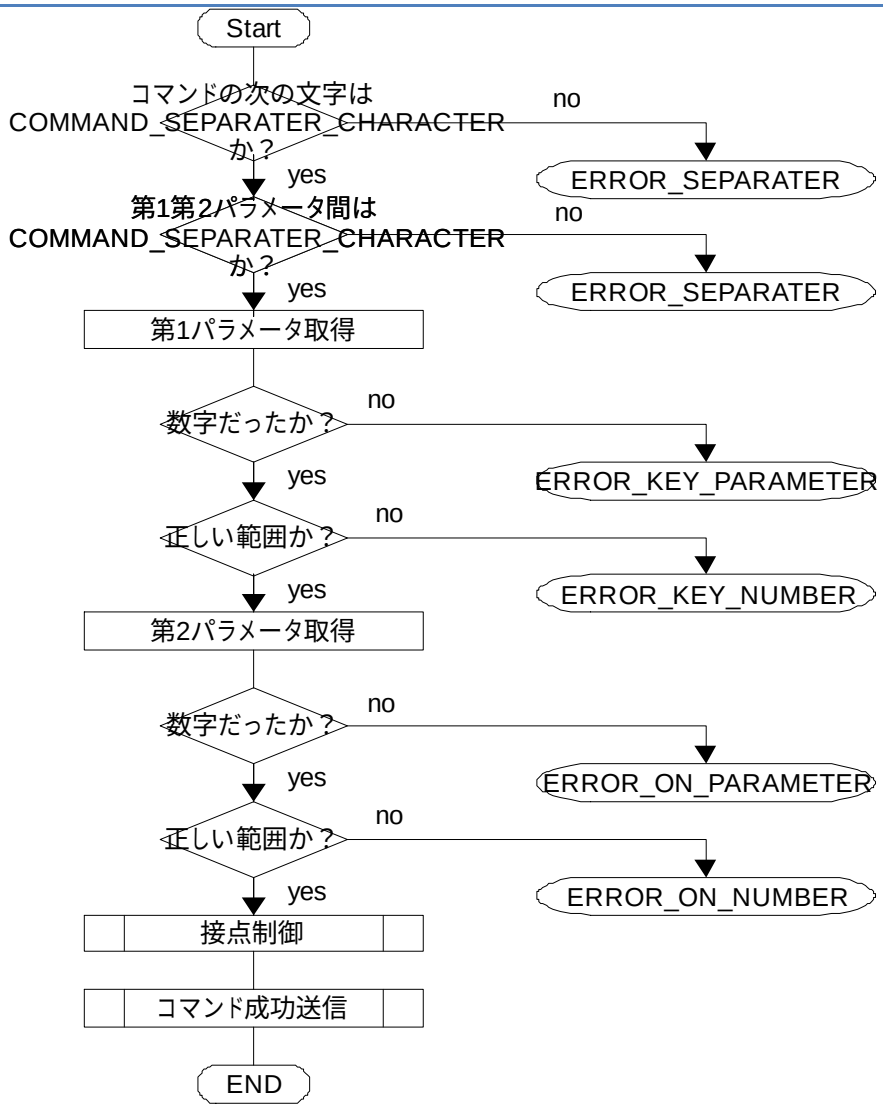
項目	内容
モジュール名	CommandCheck
定義ファイル	AutoKey.c

宣言ファイル	AutoKey.c
書式	char * CommandCheck(char *buf)
概要	受信文字列を受け取りコマンドであるかチェックします。受信文字連れがコマンドであった場合対応するコマンドを実行します。 コマンドと実効関数の定義は CList により定義されています。
引数	受信文字列のポインタ 通例 GetBufPointer 関数を使用します。
戻り値	エラーメッセージ文字列のポインタ エラーない場合は NULL が戻ります。 ERROR_NOT_COMMAND：コマンド文字列ではない ERROR_UNDEFINE_COMMAND：未定義コマンド その他各コマンドのエラーメッセージ
処理	

FuncKey

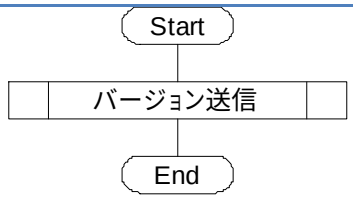
項目	内容
モジュール名	FuncKey
定義ファイル	AutoKey.c
宣言ファイル	AutoKey.c
書式	char * FuncKey(char * arg)
概要	接点信号をコマンドに従って制御します
引数	char * arg：コマンド文字列（!を除いた文字列）
戻り値	NULL：エラーなし ERROR_SEPARATER：セパレータエラー ERROR_KEY_NUMBER：キー番号指定エラー ERROR_ON_PARAMETER：第 2 パラメータ指定エラー ERROR_ON_NUMBER：第 2 パラメーター範囲エラー

処理



FuncVer

項目	内容
モジュール名	FuncVer
定義ファイル	AutoKey.c
宣言ファイル	AutoKey.c
書式	char * FuncVer(char * arg)
概要	バージョンを返す 接続確認としても使用可能
引数	char * arg：コマンド文字列（!を除いた文字列）
戻り値	NULL ※本関数は NULL 以外返さないが他のコマンドと共通化のため戻り値を設

	けている
処理	

CList

項目	内容
モジュール名	CList
定義ファイル	AutoKey.c
宣言ファイル	AuroKey.c
型	typedef struct { char Com; /* コマンドキャラクタ */ char* (*func)(char * arg); /* 戻り値エラーメッセージ 0:エラーなし*/ } CommList;

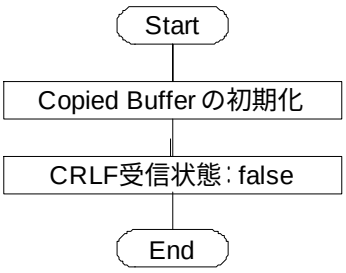
定義	const CommList CList[] = {
	{(char)'V', FuncVer}, /* バージョン表示 */
	{(char)'K', FuncKey} /* キー制御 */
	};

・Communication Device Driver

ファイル名	モジュール	内容
Usart.c	InitUsart()	通信モジュールのソフトウェア的初期化
	InitUsartLower()	通信モジュールのソフトウェア的初期化（Driver 内部使用）
	ISR(USART_RX_vect)	受信割り込み関数
	CheckCRLF()	CRLR 受信確認関数
	GetBufPointer()	文字列の先頭位置取得
	UART_write()	USART（AVR 通信モジュール）へ１文字出力 ※本プロジェクトでは 8 ビット
	SendText_P	文字列出力 ※文字列のポインタは Flash 領域の文字列用

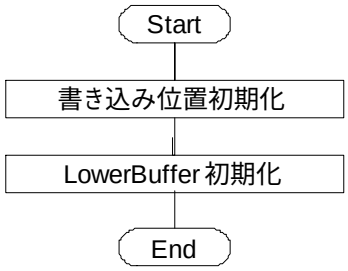
	rbuf	受信バッファ praivate 使用

InitUsart

項目	内容
モジュール名	InitUsart
定義ファイル	Usart.c
宣言ファイル	Modules.h
書式	void InitUsart(void)
概要	Main 処理用 USART 受信バッファ（copied Buffer）の初期化（CPU の USART は HartdwareInit で設定済み） 初期化 CRLF 受信状態：false 受信バッファ（copied buffer）の初期化
引数	なし
戻り値	なし
処理	

InitUsartLower

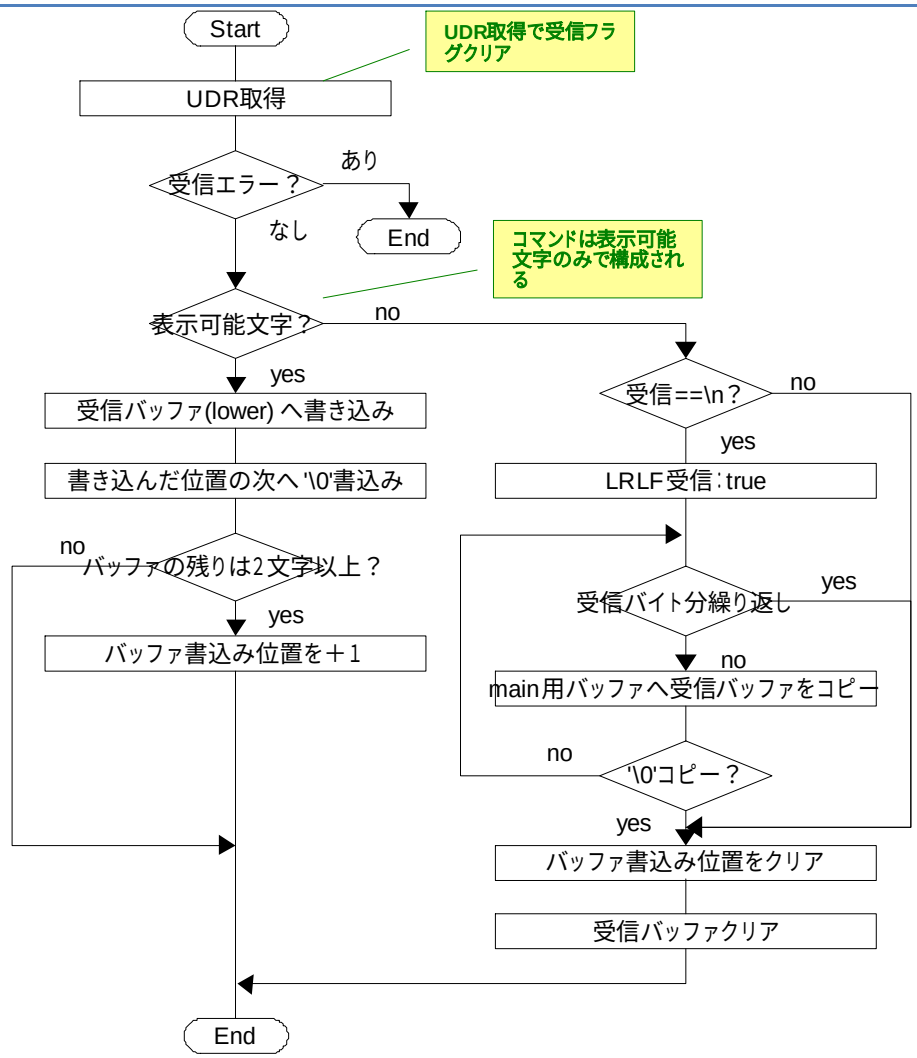
項目	内容
モジュール名	InitUsartLower
定義ファイル	Usart.c
宣言ファイル	Modules.h
書式	void InitUsartLower (void)
概要	低水準 USART 受信バッファの初期化（CPU の USART は HartdwareInit で設定済み） 初期化 受信データのバッファ書込み位置 受信バッファ（lower buffer）の初期化
引数	なし

戻り値	なし
処理	

ISR(USART_RX_vect)

項目	内容
モジュール名	ISR(USART_RX_vect)
定義ファイル	Usart.c
宣言ファイル	Usart.c
書式	ISR(USART_RX_vect)
概要	割り込み関数 割り込み要因：１キャラクタ受信完了、もしくは受信エラー
引数	なし
戻り値	なし

処理



CheckCRLF

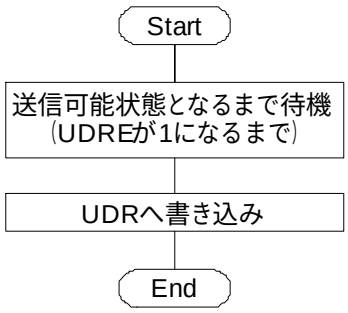
項目	内容
モジュール名	CheckCRLF
定義ファイル	Usart.c
宣言ファイル	Modules.h
書式	uint8_t CheckCRLF(void)
概要	CRLF を受信したかチェック（実際には LF の受信確認のみ） LF 以外の表示不可能キャラクターは受信処理で無視される
引数	なし
戻り値	CRLF 受信状態 true：受信状態 false：未受信

処理	

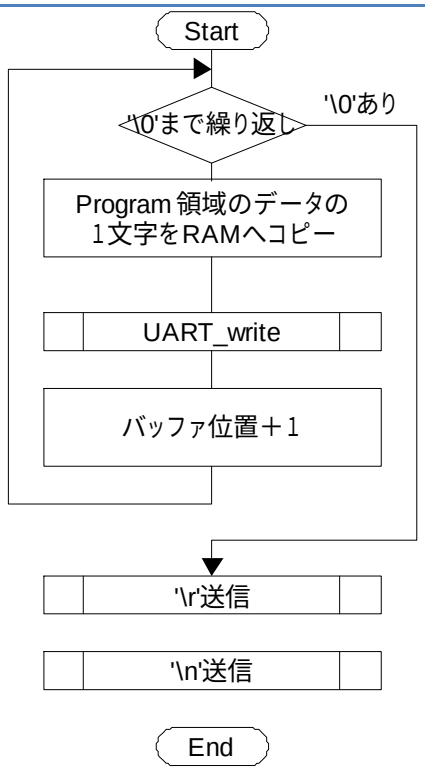
GetBufPointer

項目	内容
モジュール名	GetBufPointer
定義ファイル	Usart.c
宣言ファイル	Modules.h
書式	char * GetBufPointer(void);
概要	受信バッファ（ main 用）の先頭位置取得
引数	なし
戻り値	受信文字列の先頭位置を示すポインタ
処理	

UART_write

項目	内容
モジュール名	UART_write
定義ファイル	Usart.c
宣言ファイル	Usart.c
書式	void UART_write(char c)
概要	1 バイト USART から送信
引数	送信キャラクタ（8bit 文字）
戻り値	なし
処理	

SendText_P

項目	内容
モジュール名	SendText_P
定義ファイル	Usart.c
宣言ファイル	Modules.h
書式	void SendText_P(PGM_P buf)
概要	文字列の送信 CRLF 自動付加 送信文字列は Program 領域へ書かれたデータのポインタを指定
引数	buf：Program 領域に書かれた文字列のポインタ
戻り値	なし
処理	

rbuf

項目	内容
モジュール名	rbuf
定義ファイル	Usart.c
宣言ファイル	Usart.h
型	typedef struct { uint8_t data[16]; /* lower 受信バッファ */

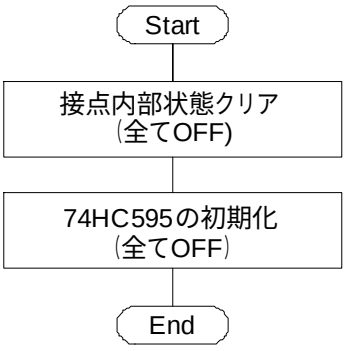
	uint8_t cdata[16]; /* main 用受信バッファ */ uint8_t pos; /* lower 受信バッファ書込み位置 */ uint8_t CRLF; /* CRLF 受信状態 true:受信完了*/ } RxBuffer;
定義	RxBuffer rbuf;

・接点制御モジュール

用途：コマンドを受信し指定されたコマンドを処理する

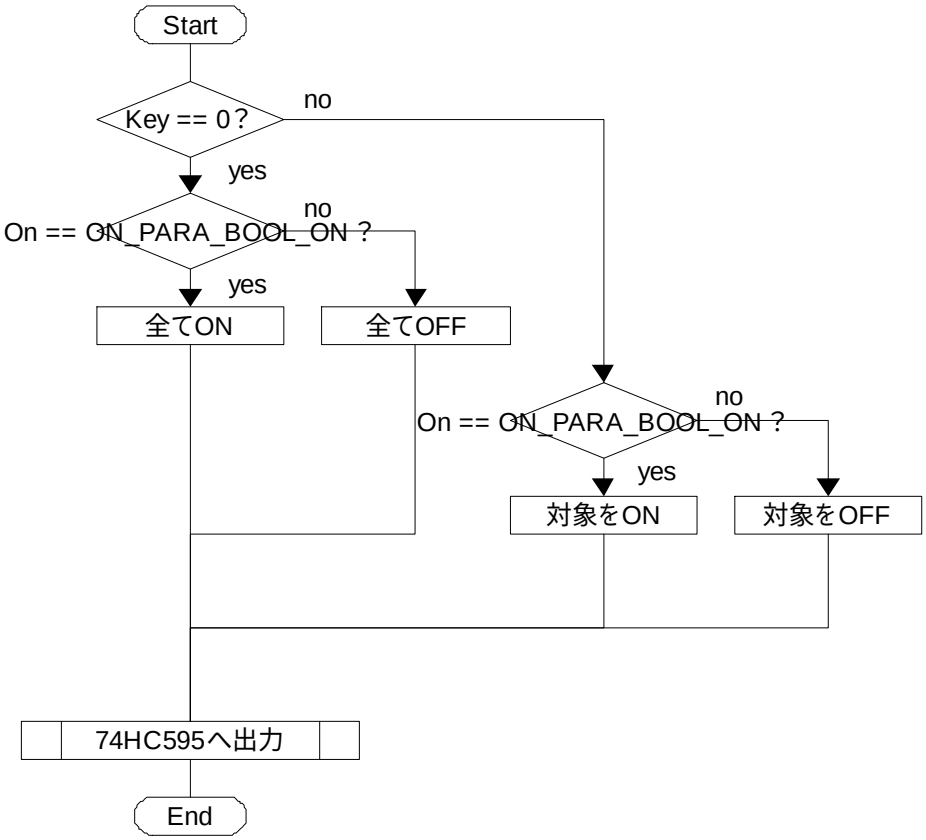
ファイル名	モジュール	内容
KeyCtrl.c	ResetKey()	キー状態のリセット（全て OFF）
	ChangeKeyStatus()	キー状態の変更
	Key keydata	キー状態の内部保持

ResetKey

項目	内容
モジュール名	ResetKey
定義ファイル	KeyCtrl.c
宣言ファイル	Modules.h
書式	void ResetKey(void)
概要	接点信号の初期化（全て OFF）
引数	なし
戻り値	なし
処理	

ChangeKeyStatus

項目	内容
モジュール名	ChangeKeyStatus
定義ファイル	KeyCtrl.c

宣言ファイル	Modules.h
書式	void ChangeKeyStatus(uint8_t Key, bool On)
概要	接点信号を制御
引数	Key：0~12 0 の場合全てのキーに対して 1～12 の場合キー番号１～１２に対して ON：0~1 0 の場合 OFF 1 の場合 ON
戻り値	なし
処理	

keydata

項目	内容
モジュール名	keydata
定義ファイル	KeyCtrl.c
宣言ファイル	Modules.h
型	typedef struct {

	union { uint16_t byte; KeyBit bit; }u; } Key;
定義	static Key keydata;

・74HC595 Device Driver

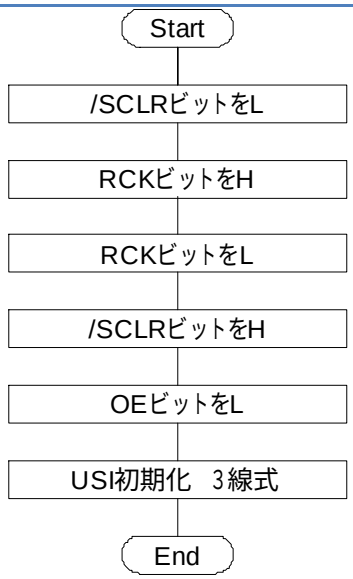
用途：コマンドを受信し指定されたコマンドを処理する

ファイル名	モジュール	内容
74HC595	ResetParaIO()	接点信号のリセット（全て OFF）
	SetParaIO	接点信号出力
	SendUsi()	USI 出力

ResetParaIO

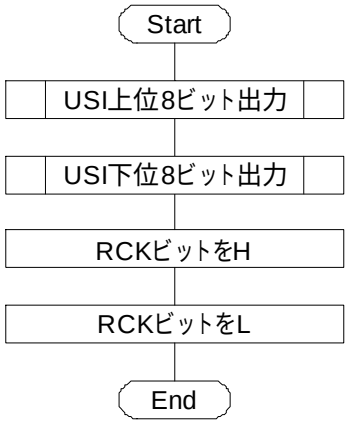
項目	内容
モジュール名	ResetParaIO
定義ファイル	74HC595.c
宣言ファイル	Modules.h
書式	void ResetParaIO(void)
概要	74HC595 の状態を初期化します（全て OFF）
引数	なし
戻り値	なし

処理



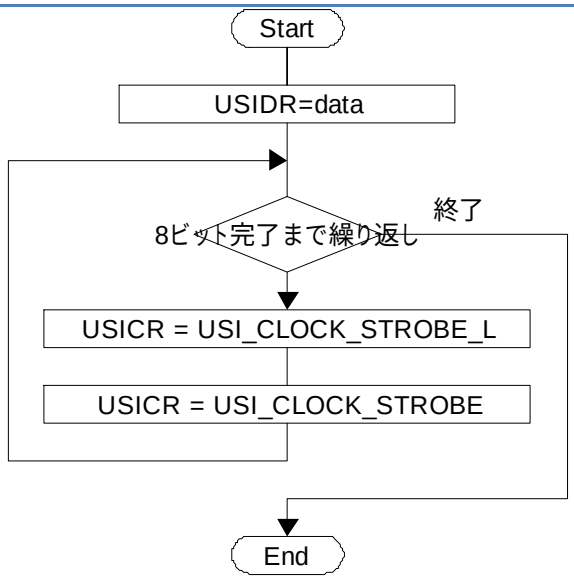
SetParaIO

項目	内容
モジュール名	SetParaIO
定義ファイル	KeyCtrl.c
宣言ファイル	Module.h
書式	void SetParaIO(uint16_t sdata)
概要	接点出力の設定
引数	sdata：USI 出力データ（16 ビット）MSB ファースト
戻り値	なし
処理	



SendUsi

項目	内容
モジュール名	SendUsi

定義ファイル	KeyCtrl.c
宣言ファイル	KeyCtrl.c
書式	void SendUsi(uint8_t data)
概要	USI（SPI）出力
引数	data：USI 出力データ（8 ビット）MSB ファースト
戻り値	なし
処理	

○コマンドフォーマット

[Comand] [para1] [Para2][CR][LF]

[Commad]：コマンド
[para1]：パラメータ 1
[para2]：パラメータ 2
[CR]：キャリッジリターン
[LF]：ラインフィード
コマンド区切り文字：スペース

○通信設定

通信方式：調歩同期式
ボーレート：38400bps
ビット長：8 ビット
ストップビット：1 ビット
フロー制御：なし

○コマンド

Command	Parameter	
!V	なし	バージョン取得
!K	[pos] [on]	[pos]：キーポジション 0~12 0：全てのキーに対して 1～12：キー１～キー１２に対して [on]：1　接点を CLOSE 0　接点を OPEN 例） !K 1 1 キー１をオン !K 0 1 全キーをオン

○コマンド応答文字列

Response	Description
!W	起動メッセージ ※リセット後 1 回のみ
!P	コマンドパス ※!K に対する応答
!V [ver]	!V（バージョン確認）に対する応答 [ver]：バージョン 例 !V 0.1
!E0	コマンドではない文字列を受信した
!E1	未定義のコマンド
!E2	!K コマンドの応答で第 1 パラメータが異常
!E3	!K コマンドの応答でキー番号が指定範囲外です
!E4	!K コマンドの応答で第 2 パラメータが異常
!E5	!K コマンドの応答で ON 状態指定が範囲以外です
!E6	!K コマンドの応答でコマンドセパレータのキャラクタが指定以外です

○マイコン

Item	Description
CPU	ATMEL 製 ATTINY2313-20PU 内蔵クロック：最大 8MHz
Flash	2k バイト
RAM	128 バイト
EEPROM	128 バイト
動作クロック	外部 Max20MHz 内蔵 Max8MHz
開発環境	AVR Studio 4 WinAVR-20080610
本プロジェクトでの使い方	内蔵クロックを使用：8MHz EEPROM は使用していません。 Flash にはプログラムの他、固定文字列データの領域として使用しています。 割り込みは RS232C の受信割り込みのみ使用 ウォッチドッグリセット：ソフトウェアモード SPI 通信：内蔵モジュール使用 空きピン：プルダウンとして入力ピン設定 書込み用のピン SCK,MISO は衝突が起きないように 100Ωの抵抗を接続する。 74HC595 との通信に USI 通信を使用します

○フィードバック

本ドキュメントならぶに本プロジェクトに関するご意見ご感想などありましたら

yama_e@users.sourceforge.jp

へご連絡ください

○プロジェクトについて

・プロジェクトサイト

http://sourceforge.jp/projects/autokey/

・メーリングリスト

http://lists.sourceforge.jp/mailman/listinfo/autokey-dev

上記 URL から購読申込みができます。

・フォーラム

http://sourceforge.jp/projects/autokey/forums/

上記サイトにフォーラムを用意しています。

・文書

本ドキュメントを含むドキュメントは下記 URL に保管しています。

http://sourceforge.jp/projects/autokey/docman/

・ソースコード（準備でき次第）

組込みソフトウェアのソースコードを提供します

下記 URL からダウンロード可能です

http://sourceforge.jp/projects/autokey/releases/

・バージョン管理（準備でき次第）

Subversion を使用します。下記 URL 参考

http://sourceforge.jp/projects/autokey/cvs/