4 IMU センサーハブの作成　　IMU4P-01 版

対象 firm: 0.01.00.201903230

0．はじめに

複数の IMU を同時に使用する場合, それぞれから出力されるセンサデータに時間的関係は不定である. そこで, それぞれのセンサからの割り込みタイミングにてタイムスタンプを付加しセンサデータとセットで扱う事でそれぞれのセンサのデータを上位アプリケーションで使用する場合, 時間的関係を考慮に入れて処理する事ができる.

1. 基本仕様

- 4 IMU の読取
- IMU I/F SPI, 8MHz, 4 線式, 4 つのチップセレクト, 4 つの割り込み入力まで
- 割り込み発生時のタイムスタンプと通算割り込み発生回数を記録
- 各センサデータ, タイムスタンプ, 通算割り込み発生回数を UART に送信
- 出力フォーマット: plain text, ハミング符号化
- UART 4MHz, 8bit, non parity, 1 stop bit
- CPU STM32L072 (32MHz)

2. 詳細仕様

2-1. ブロックダイアグラムと動作原理

図 2.1 にブロックダイヤグラムを示す.

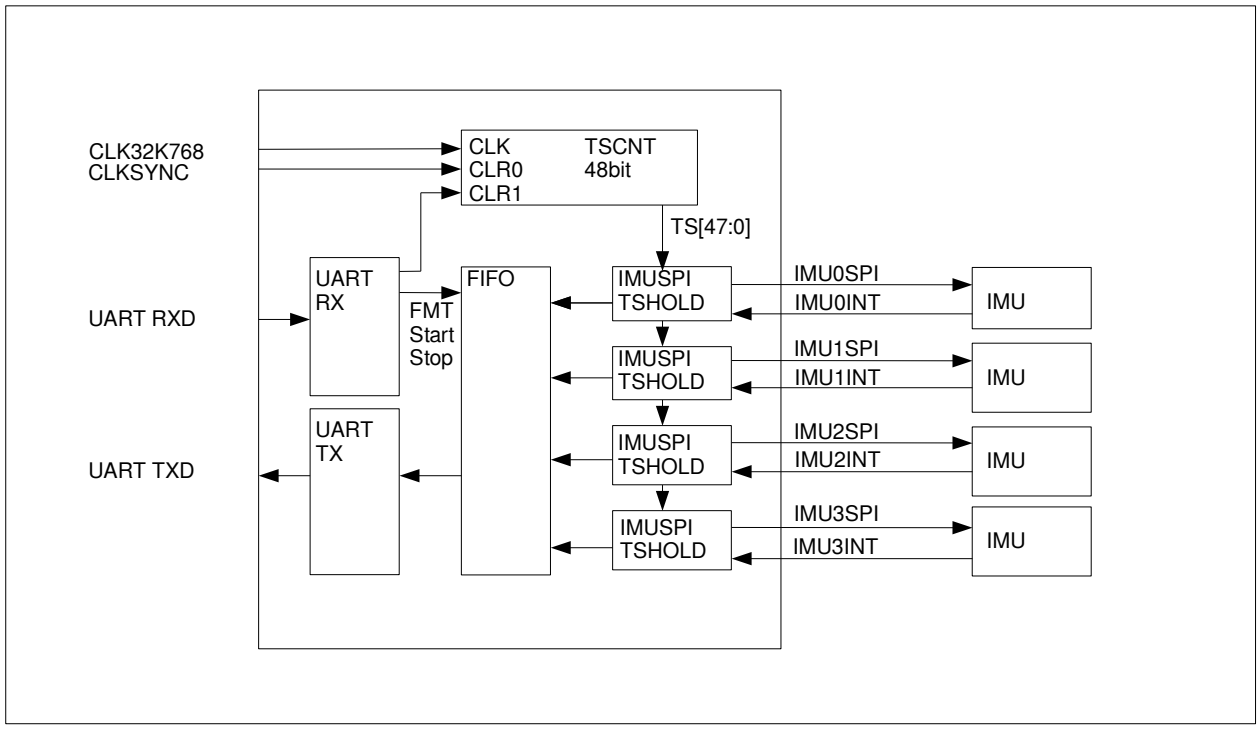


図 2.1 ブロックダイアグラム

TSCNT は, CLK32K768 の立ち上がりでカウントアップするフリーランカウンタである.
入力周波数は通常 32.768kHz を用いる. また TSCNT は値をクリアすることができる.
後述する UART RXD からの reset コマンドもしくは, CLKSYNc 端子でクリアする.

IMUSPI,TSHOLD は, IMU の初期化, データレディ割り込みの受信, タイムスタンプの保持, センサデータの取得を行う. データレディ受信すると, 割り込みが発生しその処理中にタイムスタンプを保持し, センサデータ読み込みリクエストを発行し割り込みを終了する. 通常ルーチン内で, どのセンサに対してデータの読み込みをしていなければ, 該当センサデータ読み込みリクエストがある IMU に対しセンサデータを読み込む.

FIFO は, IMUSPI,TSHOLD 段で得られたセンサデータならびにタイムスタンプを後述する UART TXD に順次転送する. この際, 指定されたデータフォーマットに変換して送信される. デフォルトはテキストフォーマットである.

UART RXD は, 外部からテキスト形式のコマンドを受け付け, 与えられたコマンドの処理を行う. 2-3 のコマンド一覧を参照の事

UART TXD は, 取得されたセンサデータならびにタイムスタンプの送信を行う. コマンドにより出力フォーマットを変更する事ができる.

2-2. 出力フォーマット

2-2-1. テキストフォーマット

図 2.2.1 にテキストフォーマットを示す

TS (6)	SPC	IMU NUM (1)	SPC	CNT (4)	SPC	CAP (4)	SPC	ACCEL (12)	SPC	GYRO (12)	SPC	CKSUM (2)	LF

TS HEX 7 文字 タイムスタンプ 28bit
IMUNUM HEX 1 文字 IMU 番号 4bit
CNT HEX 4 文字 IMU 番号内通算カウント 16bit
CAP HEX 4 文字 センサー情報有無情報 16bit
ACCEL HEX 12 文字 加速度データ 上から 4 文字ずつセンサ上の x 軸 , y 軸 , z 軸 16bit x 3
GYRO HEX 12 文字 ジャイロデータ 上から 4 文字ずつセンサ上の x 軸 , y 軸 , z 軸 16bit x 3
TEMP HEX 2 bytes 温度データ 8bit
CKSUM HEX 2 文字 チェックサム それぞれを data を 1 バイトずつ CKSUM まで XOR すると 0 になる数 8bit
SPC 空白 (0x20)
LF Linefeed (0x0a8

図 2.2.1 テキストフォーマット

2-2-2. ハミング符号フォーマット

図 2.2.2 にハミング符号フォーマットを示す

TS (7)	IMU NUM (1)	CNT (2)	CAP (4)	ACCEL (12)	GYRO (12)	TEMP (2)	CRC (2)

TS 7 bytes タイムスタンプ 28bit
IMUNUM 1 byte IMU 番号 4bit
CNT 2 bytes IMU 番号内通算カウント 8bit
CAP 4 bytes センサー情報有無情報 16bit
ACCEL 12 bytes 加速度データ 上から 4 文字ずつセンサ上の x 軸 , y 軸 , z 軸 16bit x 3
GYRO 12 bytes ジャイロデータ 上から 4 文字ずつセンサ上の x 軸 , y 軸 , z 軸 16bit x 3
TEMP 2 bytes 温度データ 8bit
CRC 2 bytes TS から TEMP までの CRC32 の計算をした下位 1 バイト 8bit7

q[3:0] = d[3:0]
q[4] = d[1] ^ d[2] ^ d[3]
q[5] = d[0] ^ d[2] ^ d[3]
q[6] = d[0] ^ d[1] ^ d[3]
q[7] = frame start bit

図 2.2.2 ハミング符号フォーマット

2-2-3. CAP ビット

CAP ビットは本機から送信されるデータの有無を示したものである. TS, IMUNUM, CNT に続くデータは, CAP bit の小さい番号から順に並べられる. CAP の bit 番号とそれに定義されるセンサー情報を表 2.2.3 に, また ACCEL, GYRO, TEMP, CRC32_8 のビットが立っていた時の例を図 2.2.3 に示す.

ビット	名前	内容
15	EXTENSION	拡張ビット (TBD)
14	CRC32_8	CRC32 が有効 ただし, 下位 8 ビットのみ
13	CRC32	CRC32 が有効
12 – 4		(Reserved)
3	TEMP	温度センサーが有効　(1byte 0: -20deg, 255: 107.5deg 1LSB=0.5deg)
2	MAGNETICS	磁気センサーが有効　(3 軸 各 2 bytes)
1	GYRO	ジャイロセンサーが有効 (3 軸 各 2 bytes)
0	ACCEL	加速度センサーが有効 (3 軸 各 2 bytes)

表 2.2.3 センサー情報有無情報

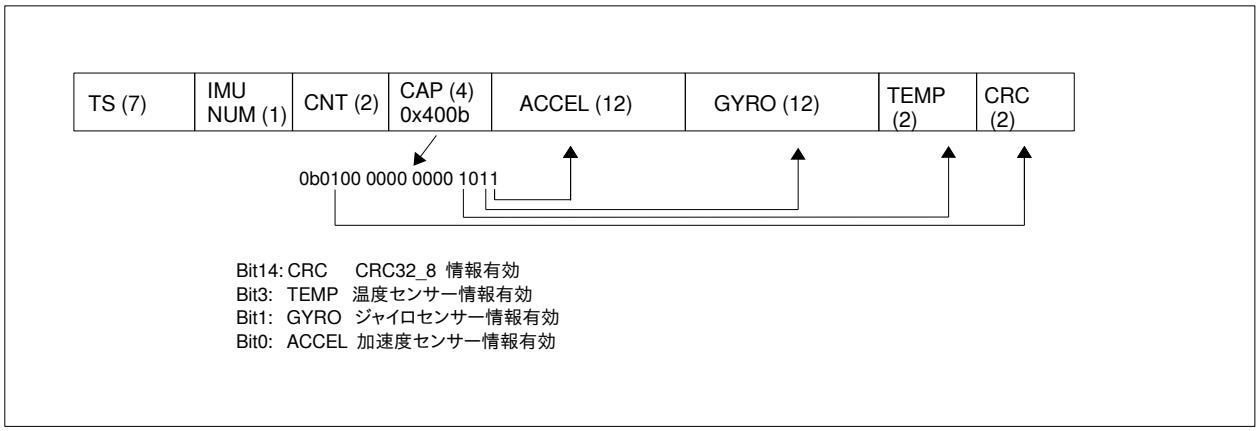


図 2.2.3 CAP ビットとセンサー情報の並びの例

2-3. コマンド一覧

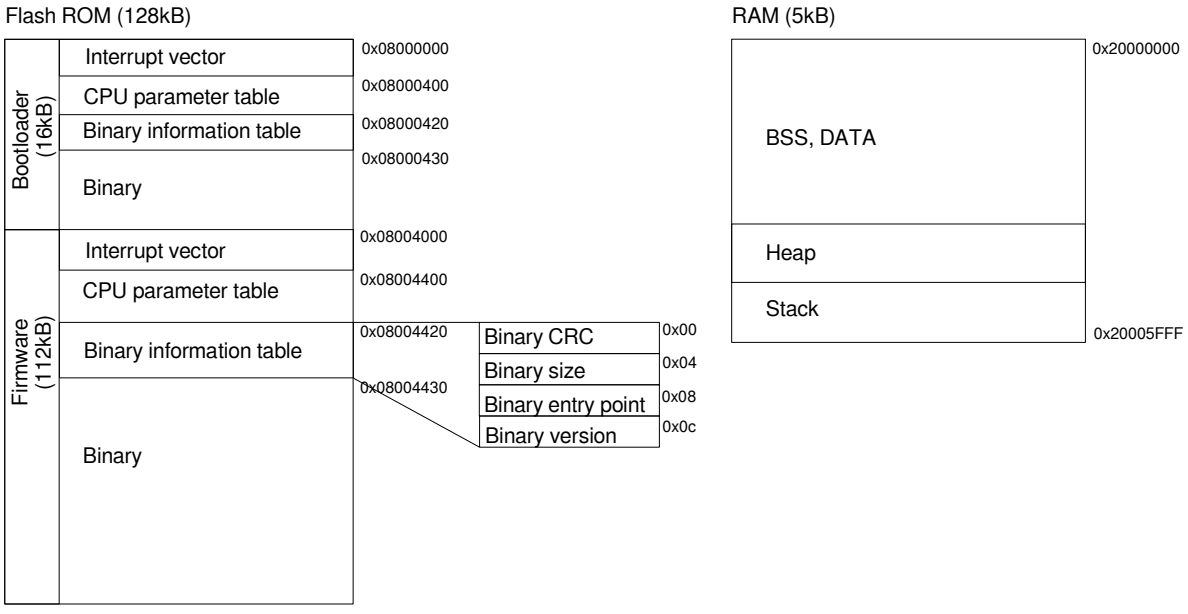
format [num]	IMU センサーデータのフォーマット変更 [num] IMU 番号
start	センサーデータの送信開始
stop	センサーデータの送信停止
reset	タイマースタンプのカウンタークリア
init	IMU 設定変更後の初期化
imu [num] set [freq] [acc] [gyro] [mag]	IMU 設定 [num] IMU 番号 [freq] frequency in Hz [acc] accelation in 10mG [gyro] gyro in 10mDPS [mag] magnetics in mT
imu [num] get	IMU 設定確認 [num] IMU 番号

imu [num] regset [reg] [val]	IMU の レジスタ設定 [num] IMU 番号 [reg] 対象レジスタ [val] 設定値
imu [num] regget [reg] [cnt]	IMU の レジスタ値の読込み [cnt] 読込み数 8 文字まで
id set [str]	ID の設定 [str] ID の文字列 32 文字まで
id get	ID の表示

2-4. ピン配置

信号名	STM32L072 ピン名	IO	内容
IMUINT0	PA0/TIM2IC0	I	IMU0 センサーデータ取得割り込み
IMUCS0X	PA13	O	IMU0 SPI チップセレクト
IMUINT1	PA1/TIM2IC1	I	IMU1 センサーデータ取得割り込み
IMUCS1X	PA14	O	IMU1 SPI チップセレクト
IMUINT2	PA2/TIM2IC2	I	IMU2 センサーデータ取得割り込み
IMUCS2X	PB0	O	IMU2 SPI チップセレクト
IMUINT3	PA3/TIM2IC3	I	IMU3 センサーデータ取得割り込み
IMUCS3X	PB1	O	IMU3 SPI チップセレクト
SPI1SCK	PA4/SPI1SCK	O	SPI クロック
SPI1MOSI	PA6/SPI1MOSI	O	SPI MOSI
SPI1MISO	PA5/SPI1MISO	I	SPI MISO
CLK32K768IN	PA15/TIM2ETR	I	Timestamp カウンタークロック入力
CLKSYNC	PA4	I	Timestamp カウンタークリア信号 (TBD)
UARTTXD	PA9/USART1TXD	O	センサーデータ出力
UARTRXD	PA10/USART1RXD	I	コマンド入力

2-5. メモリマップ



0x0801FFFF

3. ソフトウェア

3.1 ファームウェアの書き込み

STM32L072 の BOOT ピンを high にした状態で USB を Ubuntu PC に接続する Ubuntu に接続し, lsusb で

ID 0483:df11 STMicroelectronics STM Device in DFU Mode

がある事を確認する. もし, ないようなら USB の接続がおかしいと思われる.

問題なく ID が見えていれば, 各ファームウェアを用意し以下のコマンドを実行する

% dfu-util -D KICAD.bootfirm.yyyymmdds.bin -a 0 --dfuse-address 0x8000000

ここで，dfu-util がない場合は, 以下の通り apt install すると良い

% sudo apt install dfu-util

dfu-util を動作させ以下の通り問題なく success が出れば, USB ケーブルを PC から抜き再度挿し直しをおこなう

% dfu-util -D imu4p.x.yy.zz.yyyymmdds.bootfirm.bin -a 0 --dfuse-address 0x8000000
dfu-util 0.9

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

dfu-util: Invalid DFU suffix signature
dfu-util: A valid DFU suffix will be required in a future dfu-util release!!!
Opening DFU capable USB device...
ID 0483:df11
Run-time device DFU version 011a
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 2048
DfuSe interface name: "Internal Flash "
dfu-util: Non-valid multiplier 'g', interpreted as type identifier instead
Downloading to address = 0x08000000, size = 75264
Download [=========================] 100%   75264 bytes
Download done.
File downloaded successfully

ただしサイズは firmware の version によって変化するので注意が必要である

4. 制限事項

4-1. 未実装

なし

4-2. 開発環境

ファームウェア開発環境は ubuntu + make + gcc

5. ソフトウェアの改変と回路図

https://github.com/zhtlab/McuDevelop/tree/master/examples/IMU4P