IMU6

Inertial Measurement Unit

▶ PRODUCT DESCRIPTION

IMU6 is specially designed to replace STIM300, it is compatible completely with STIM300 in both hardware and software.

IMU6 is an IMU consisting of 3 high accuracy MEMS-based gyros, 3 high stability accelerometers and 3 high stability inclinometers in a miniature package. Each axis is factory-calibrated for bias, sensitivity and compensated for temperature effects to provide high-accuracy measurements in the temperature range -40°C to +85°C. The unit runs off a single +5V supply.

IMU6 communicates via a standard high-level RS422 interface. The use of a 32-bit RISC ARM microcontroller provides flexibility in the configuration, like choice of output unit, sample rate, low pass filter –3dB frequency and RS422 bit-rate and protocol parameters. All configurable parameters can be defined when ordering or set by customer. When IMU6 is powered up, it will perform an internal system check and synchronise the sensor channels. As an acknowledgement of the complete power-up sequence, it will provide special datagrams containing part number, serial number and configuration data. IMU6 will then automatically proceed to provide measurement data. Connect power and IMU6 will provide accurate measurements over the RS422 interface. The measurement data is transmitted as packages of data on a fixed format (datagram) at intervals given by the sample rate together with a synchronization signal (TOV). The datagram is in binary coded format in order to have an efficient transfer of data. In addition to the measurement data itself, the datagram contains an identifier, status bytes and a 32 bit CRC (Cyclic Redundancy Check) to provide high degree of fault detection in the transmissions. The status bytes will flag any detected errors in the system. IMU6 can also be configured to transmit data only when triggered by a separate digital input signal (ExtTrig).

For more advanced users, the gyro may be put in Service Mode. In this mode all the configuration parameters can be intermediately or permanently changed by overwriting the current settings in the flash memory. In Service Mode the commands and responses are in a human readable format (ASCII); to enable the use of terminal-type software during typical product integration. Service Mode also provides the ability to perform single measurements, perform diagnostics and obtain a higher detail level of detected errors reported in the status bytes.

PM Instrumentation | 59 rue Emile Deschanel | F-92400 Courbevoie | France
+33(0)1 46 91 93 32 | contact@pm-instrumentation.com | www.pm-instrumentation.com

▶ TECHNICAL DATA

Parameters		Test condition	Min value	Typical value	Max value	Unit
Gyro	Dynamic range			±400		º/s
	Zero bias stability			5		º/h
	Zero bias in the full temperature range difference	The rms value of the zero error of the fixed temperature point relate to full temperature from -45 ℃ to 60℃		10		°/h
	Random walk			0.2		º/√h
	bias acceleration sensitivity			1		º/h /g
	Scale factor nonlinearity	FS=400 º/s		15		ppm
	Bandwidth			200		Hz
accelerometer	Dynamic measurement range			±30		g
	Zero bias stability			0.15		mg
	Zero bias error over the full temperature range	-45°C~60°C, variable temperature ΔT ≤±1°C/min, rms value		6		mg
	Random walk			0.2		m/s/√h
	Scale factor nonlinearity	FS=10g		300		ppm
	Bandwidth			100		Hz
Electrical characteristics	Voltage	DC	5.0			V
	Consumption			5		W
	Ripple	P-P			100	mV
Use environment	Working temperature		-45		65	℃
	Store temperature		-60		85	℃
	Vibration			10~2000Hz, 6.06g		
	Impact			5000g, 0.1ms		
Reliability	MTBF			20000		h
	Continuous working time			120		h

▶ ELECTRICAL INTERFACE DIMENSION

The electrical connector type is J30J-15ZKN-J, and the specific assignment of the contacts is shown in Table 1 below.

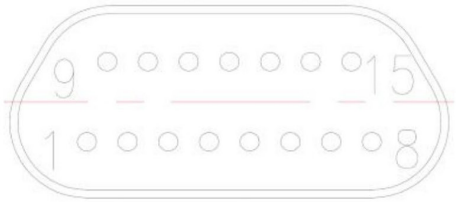
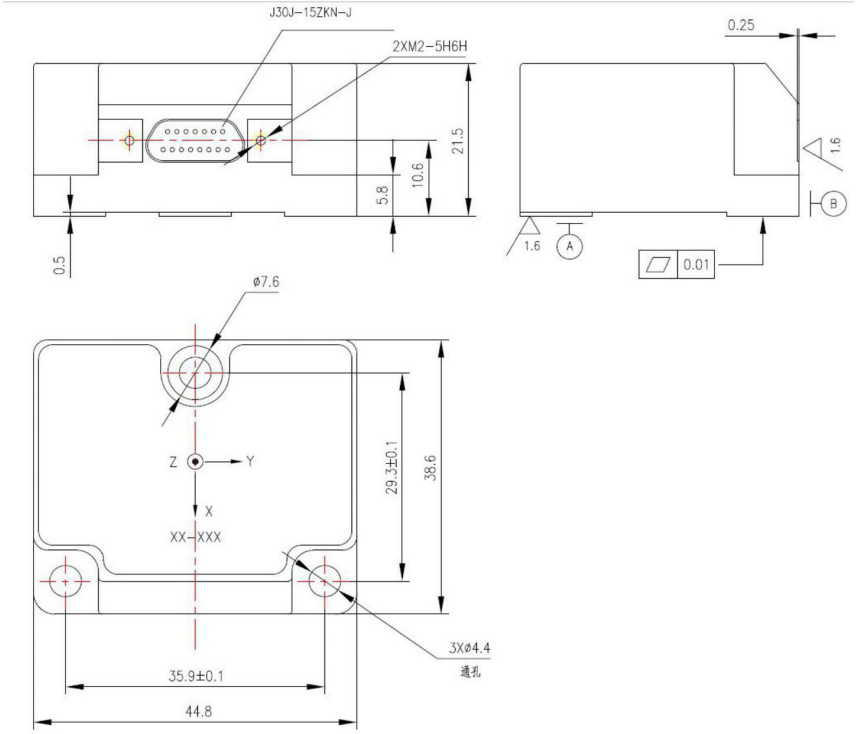


Table 1 J30J-15ZKN-J contact distribution

Node number	Definition	Usage
10	Rx+	RS422
2	Rx-	
1	Tx-	
9	Tx+	
8	+5V	Power +
15	GND	Power ground
other		Factory use, reserve

▶ STRUCTURE SIZE AND COORDINATE DEFINITION



▶ SERIAL PORT COMMUNICATION PROTOCOL

The interface adopts RS422 communication protocol, the baud rate is 921600bps, 1 start bit, 8 data bits, 1 stop bit, no parity. The packet format is as follows in Table 1 and Table 2:

Table 1 Product Output Packets

ID	0xA5
gyro	3x3Byte
Gyro status	1Byte
accelerometer	3x3Byte
Accelerometer status	1Byte
Gyro temp.	3x2Byte
Gyro temp. status	1Byte
Accelerometer temp.	3x2Byte
Accelerometer temp. status	1Byte
Packet count	1Byte
Latency	2Byte
CRC32	4Byte

Table 2 Product output data format definition

No.	Parameter	Effective range	Byte	Scale	Remark
1	Frame head	0xA5	1	--	Packet header
2	X angular speed	[-400，400]	3	2^{-14}	Unit: °/s, first high and then low, the highest bit of the first byte is the sign bit. See the specific algorithm in description 1
3	Y angular speed	[-400，400]	3	2^{-14}	
4	Z angular speed	[-400，400]	3	2^{-14}	
5	Gyro status	--	1	--	All 0 is normal, as defined in Table 3.
6	X acceleration	[-30，30]	3	2^{-18}	Unit: g, first high and then low, the highest bit of the first byte is the sign bit. See the specific algorithm in description 3
7	Y acceleration	[-30，30]	3	2^{-18}	
8	Z acceleration	[-30，30]	3	2^{-18}	
9	Accelerometer status	--	1	--	All 0 is normal, as defined in Table 3.
10	X gyro temp.	[-128，128]	2	2^{-8}	Unit: °C, first high and then low, the highest bit of the first byte is the sign bit. See the specific algorithm in description 2.
11	Y gyro temp.	[-128，128]	2	2^{-8}	
12	Z gyro temp.	[-128，128]	2	2^{-8}	
13	Gyro temp. status	--	1	--	All 0 is normal, as defined in Table 3.
14	X accelerometer temp.	[-128，128]	2	2^{-8}	Unit: °C, first high and then low, the highest bit of the first byte is the sign bit. See the specific algorithm in description 2.
15	Y accelerometer temp.	[-128，128]	2	2^{-8}	
16	Z accelerometer temp.	[-128，128]	2	2^{-8}	
17	Accelerometer temp. status	--	1	--	All 0 is normal, as defined in Table 3.
18	Frame count	[0，255]	1	1	0-255 continuous counting
19	Delay		2		Unit: us, first high and then low, the highest bit of the first byte is the sign bit. See the specific algorithm in description 4
20	CRC32	--	4	--	CRC32 check, see description 5

Description:

1) Gyro angular velocity output [°/s]= $\frac{AR_1 \cdot 2^{16} + AR_2 \cdot 2^8 + AR_3}{2^{14}}$ ata bit format is shown in Figure 1;

AR_1 is the upper eight bits in the three bytes of each gyro axis angular speed output;

AR_2 is the middle eight bits in the three bytes of each gyro axis angular speed output;

AR_3 is the lower eight bits in the three bytes of each gyro axis angular speed output;

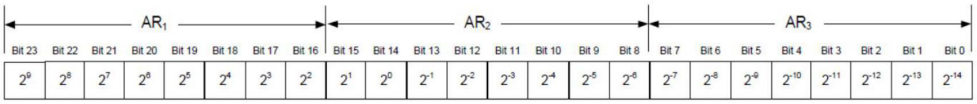


Figure 1 Converting the gyro angular velocity output to [°/s]

2) Temperature output [°C]= $\frac{T_1 \cdot 2^8 + T_2}{2^8}$)e data bit format is shown in Figure 2.

T_1 is the upper eight bits in the two bytes of each axis temperature output;

T_2 is the lower eight bits in the two bytes of each axis temperature output;

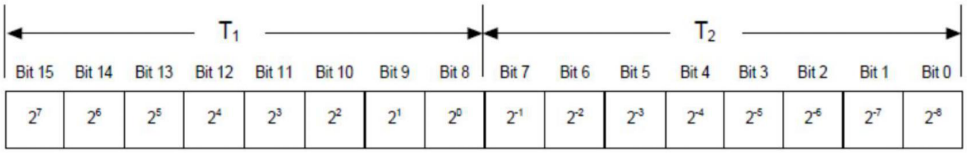


Figure 2 converts the temperature output to [°C]

3) acceleration speed output [g]= $\frac{AR_1 \cdot 2^{16} + AR_2 \cdot 2^8 + AR_3}{2^{18}}$

AR_1 is the upper eight bits in the three bytes of each accelerometer axis angular speed output;

AR_2 is the middle eight bits in the three bytes of each accelerometer axis angular speed output;

AR_3 is the lower eight bits in the three bytes of each accelerometer axis angular speed output;

4) Delay time output [us]= $T_1 \cdot 2^8 + T_2$

T_1 the upper eight bits in the two bytes of delay time output;

T_2 the upper eight bits in the two bytes of delay time output;

5) CRC32 check

The CRC32 checksum of all bytes before the CRC byte, the CRC32 check uses the following formula:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

seed = 0xFFFFFFFF

The CRC32 verification procedure is as follows

```
/************************************************************
CRC data sheet
************************************************************/
static Uint32 crc_table[256]={
0x00000000, 0x04c11db7, 0x09823b6e, 0x0d4326d9, 0x130476dc, 0x17c56b6b, 0x1a864db2,
0x1e475005, 0x2608edb8, 0x22c9f00f, 0x2f8ad6d6, 0x2b4bcb61, 0x350c9b64, 0x31cd86d3,
0x3c8ea00a, 0x384fbdbd, 0x4c11db70, 0x48d0c6c7, 0x4593e01e, 0x4152fda9, 0x5f15adac,
0x5bd4b01b, 0x569796c2, 0x52568b75, 0x6a1936c8, 0x6ed82b7f, 0x639b0da6, 0x675a1011,
0x791d4014, 0x7ddc5da3, 0x709f7b7a, 0x745e66cd, 0x9823b6e0, 0x9ce2ab57, 0x91a18d8e,
0x95609039, 0x8b27c03c, 0x8fe6dd8b, 0x82a5fb52, 0x8664e6e5, 0xbe2b5b58, 0xbaea46ef,
0xb7a96036, 0xb3687d81, 0xad2f2d84, 0xa9ee3033, 0xa4ad16ea, 0xa06c0b5d, 0xd4326d90,
0xd0f37027, 0xddb056fe, 0xd9714b49, 0xc7361b4c, 0xc3f706fb, 0xceb42022, 0xca753d95,
0xf23a8028, 0xf6fb9d9f, 0xfbb8bb46, 0xff79a6f1, 0xe13ef6f4, 0xe5ffeb43, 0xe8bccd9a, 0xec7dd02d,
0x34867077, 0x30476dc0, 0x3d044b19, 0x39c556ae, 0x278206ab, 0x23431b1c, 0x2e003dc5,
```

```
0x2ac12072,  0x128e9dcf,  0x164f8078,  0x1b0ca6a1,  0x1fcdbb16,  0x018aeb13,  0x054bf6a4,
0x0808d07d,  0x0cc9cdca,  0x7897ab07,  0x7c56b6b0,  0x71159069,  0x75d48dde,  0x6b93dddb,
0x6f52c06c,  0x6211e6b5,  0x66d0fb02,  0x5e9f46bf,  0x5a5e5b08,  0x571d7dd1,  0x53dc6066,
0x4d9b3063,  0x495a2dd4,  0x44190b0d,  0x40d816ba,  0xaca5c697,  0xa864db20,  0xa527fdf9,
0xa1e6e04e,  0xbfa1b04b,  0xbb60adfc,  0xb6238b25,  0xb2e29692,  0x8aad2b2f,  0x8e6c3698,
0x832f1041, 0x87ee0df6, 0x99a95df3, 0x9d684044, 0x902b669d, 0x94ea7b2a,
0xe0b41de7, 0xe4750050, 0xe9362689, 0xedf73b3e, 0xf3b06b3b, 0xf771768c, 0xfa325055, 0xfef34de2,
0xc6bcf05f, 0xc27dede8, 0xcf3ecb31, 0xcbffd686, 0xd5b88683, 0xd1799b34, 0xdc3abded, 0xd8fba05a,
0x690ce0ee,  0x6dcdfd59,  0x608edb80,  0x644fc637,  0x7a089632,  0x7ec98b85,  0x738aad5c,
0x774bb0eb,  0x4f040d56,  0x4bc510e1,  0x46863638,  0x42472b8f,  0x5c007b8a,  0x58c1663d,
0x558240e4,  0x51435d53,  0x251d3b9e,  0x21dc2629,  0x2c9f00f0,  0x285e1d47,  0x36194d42,
0x32d850f5, 0x3f9b762c, 0x3b5a6b9b, 0x0315d626, 0x07d4cb91, 0x0a97ed48, 0x0e56f0ff, 0x1011a0fa,
0x14d0bd4d,  0x19939b94,  0x1d528623,  0xf12f560e,  0xf5ee4bb9,  0xf8ad6d60,  0xfc6c70d7,
0xe22b20d2, 0xe6ea3d65, 0xeba91bbc, 0xef68060b,
0xd727bbb6,  0xd3e6a601,  0xdea580d8,  0xda649d6f,  0xc423cd6a,  0xc0e2d0dd,  0xcda1f604,
0xc960ebb3,  0xbd3e8d7e,  0xb9ff90c9,  0xb4bcb610,  0xb07daba7,  0xae3afba2,  0xaafbe615,
0xa7b8c0cc,  0xa379dd7b,  0x9b3660c6,  0x9ff77d71,  0x92b45ba8,  0x9675461f,  0x8832161a,
0x8cf30bad,  0x81b02d74,  0x857130c3,  0x5d8a9099,  0x594b8d2e,  0x5408abf7,  0x50c9b640,
0x4e8ee645, 0x4a4ffbf2, 0x470cdd2b, 0x43cdc09c, 0x7b827d21, 0x7f436096, 0x7200464f, 0x76c15bf8,
0x68860bfd,  0x6c47164a,  0x61043093,  0x65c52d24,  0x119b4be9,  0x155a565e,  0x18197087,
0x1cd86d30,  0x029f3d35,  0x065e2082,  0x0b1d065b,  0x0fdc1bec,  0x3793a651,  0x3352bbe6,
0x3e119d3f, 0x3ad08088, 0x2497d08d, 0x2056cd3a, 0x2d15ebe3, 0x29d4f654,
0xc5a92679,  0xc1683bce,  0xcc2b1d17,  0xc8ea00a0,  0xd6ad50a5,  0xd26c4d12,  0xdf2f6bcb,
0xdbee767c, 0xe3a1cbc1, 0xe760d676, 0xea23f0af, 0xeee2ed18, 0xf0a5bd1d, 0xf464a0aa, 0xf9278673,
0xfde69bc4,  0x89b8fd09,  0x8d79e0be,  0x803ac667,  0x84fbdbd0,  0x9abc8bd5,  0x9e7d9662,
0x933eb0bb,  0x97ffad0c,  0xafb010b1,  0xab710d06,  0xa6322bdf,  0xa2f33668,  0xbcb4666d,
0xb8757bda, 0xb5365d03, 0xb1f740b4
};
                        /*****************************************************************

Perform CRC32 calculation on the transmitted packet
Calculates the CRC-32 of a block of data all at once
Use the exhaustive list to get crc32.
Uint16 *pch is an array that needs to calculate CRC32
Intl en is the length of the array
#define POLY 0x04C11DB7 // CRC32 generator polynomial Normal
// x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x+ 1
Such as: X32+X26+...X1+1, poly=(1<<26)|...|(1<<1)|(1<<0)
*****************************************************************/
void CRC32(Uint16 *pch,int len)
{
Uint32 reg = 0xFFFFFFFF; // Initial value
int i;
for( i = 0; i < len; i++)
{
```

```
reg = (reg<<8) ^ crc_table[(((reg>>24)&0xFF) ^ pch[i])];
}
crc_data[0] = (reg>>24) & 0xFF;
crc_data[1] = (reg>>16) & 0xFF;
crc_data[2] = (reg>>8) & 0xFF;
crc_data[3] = reg & 0xFF;
return;
}
```

6, self-test function and working status real-time output function

The product has a self-test function and a real-time output function of the working status. The output data packet defined in Section 5 contains a byte indicating the status, which is powered on.

After the startup is completed, the product working status information is output in real time. The status bits are defined as shown in Table 3.

Table 3 Product Status Bit Definitions

Bit	Definition
7	0 = normal, 1 = full system exception
6	0=Normal, 1=Starting
5	0=Normal, 1=External environment exception
4	0 = normal, 1 = three axes out of use conditions
3	0=Normal, 1=Three-axis output has error
2	0=Normal, 1=Z axis is out of use condition or error
1	0=Normal, 1=Y axis exceeds usage conditions or error
0	0=Normal, 1=X axis is out of use condition or error