

6축 힘/토크 센서

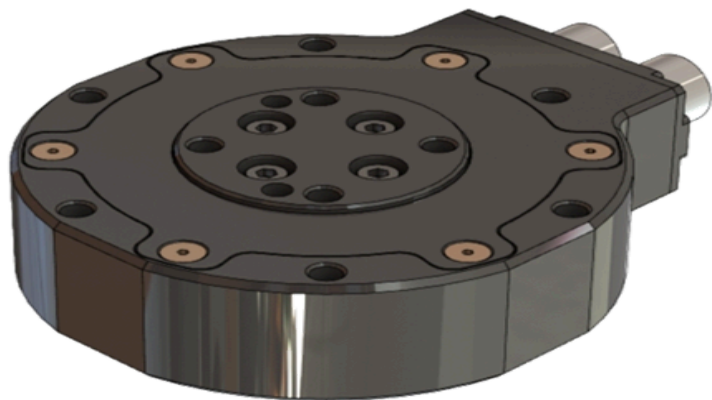
6축 힘/토크 센서 AFT200-D80-EC (KOR)

Rev. 2026.09.15

AIDIN
ROBOTICS

Force/Torque sensors user manual

This document is a user manual for AIDIN ROBOTICS
Force/Torque sensors



매뉴얼 안내

저희 힘/토크 센서를 구매해 주셔서 감사합니다.

이 매뉴얼에는 AIDIN ROBOTICS AFT200-D80 센서를 올바르게 사용하기 위한 필수 정보가 포함되어 있습니다.

소프트웨어 사용 시 반드시 본 매뉴얼을 자세히 확인해주시기 바랍니다.

로봇 시스템이 규정된 조건 밖에서 사용될 경우, 제품의 기본 성능이 발휘되지 않을 수 있음을 알려드립니다.

이 매뉴얼은 힘/토크 센서 사용 중 발생할 수 있는 위험과 그로 인한 결과에 대해 설명하고 있습니다.

안전하고 올바른 사용을 위해 반드시 본 매뉴얼에 명시된 안전 수칙을 준수하시기 바랍니다.

공지

본 매뉴얼은 AIDIN ROBOTICS의 허가 없이 복사, 재생산, 공유를 엄격히 금지합니다.

매뉴얼 또는 제공된 지침에서 오류를 발견하시면 저희에게 반드시 알려주시기 바랍니다.

(연락처: Sales@aidinrobotics.co.kr)

제조사

주식회사 에이딘로보틱스

안전 수칙

 이 기호는 관련 지침을 제대로 따르지 않을 경우 심각한 부상 또는 사망 위험이 있음을 나타냅니다.

 이 기호는 관련 지침을 제대로 따르지 않을 경우 인명 피해나 장비 및 시설의 물리적 손상 위험이 있음을 나타냅니다.

 AIDIN ROBOTICS의 AFT200-D80-EC 센서는 CE 인증 "A 등급"을 획득하였습니다.

1. 개요

1.1 6축 힘/토크 센서 AFT200-D80-EC



1.2 주요 특징

- 정전용량 방식의 스마트 6축 힘/토크 센서
- 올인원 센서 (추가 증폭기 불필요)
- 디지털 출력 통신 (EtherCAT)
- 간편한 설치 및 데이터 수집
- 그리퍼, 로봇 손, 협동 로봇, 산업용 로봇에 적용 가능

1.3 사양

Index	Unit	Value
Operating voltage	VDC	12
Max. safe excitation voltage	VDC	24
Nominal force range	N	200

Index	Unit	Value
Nominal torque range	Nm	15
Limit force (Fxyz)	N	300
Limit torque	Nm	22.5
Force Resolution	N	0.15
Torque Resolution	Nm	0.015
Force Noise-free resolution (STD)	N	0.4
Torque Noise-free resolution (STD)	Nm	0.025
Maximum sample rate	Hz	1,000
Dimensions	mm	D80 x H21.5
IP rating		IP65
Operating temperature		10-50°C

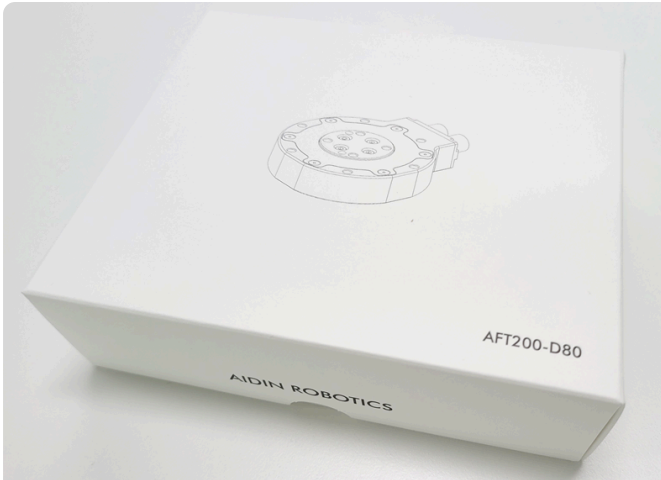
1.4 Flyer

- 📎 [AIDIN ROBOTICS] Mini_AFT200 Leaflet(KO).pdf (파일 준비 중)

2. 설치 가이드

2.1 Basic Components

- AFT200-D80 x 1 EA
- 1.5m 케이블 x 1 EA



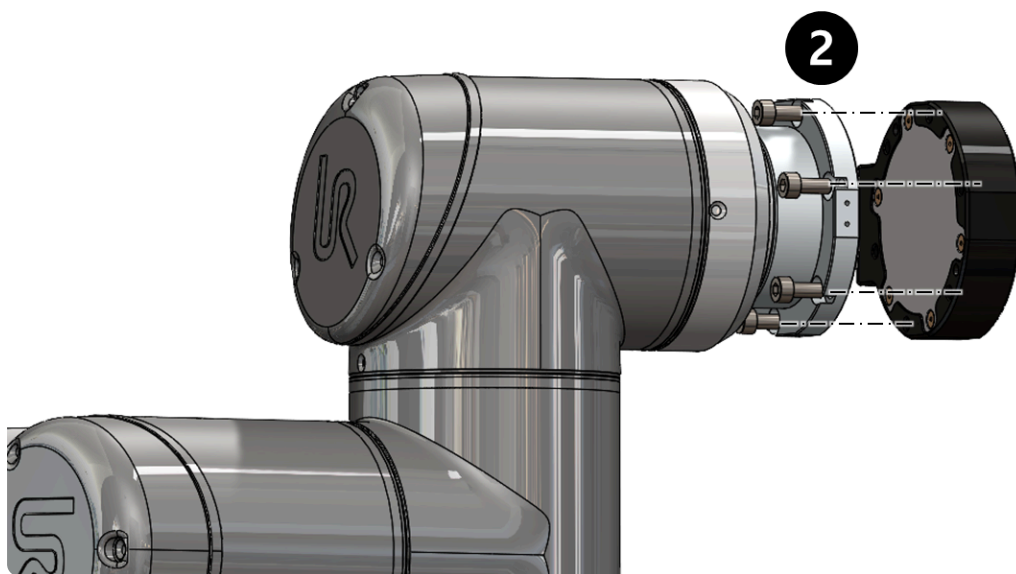
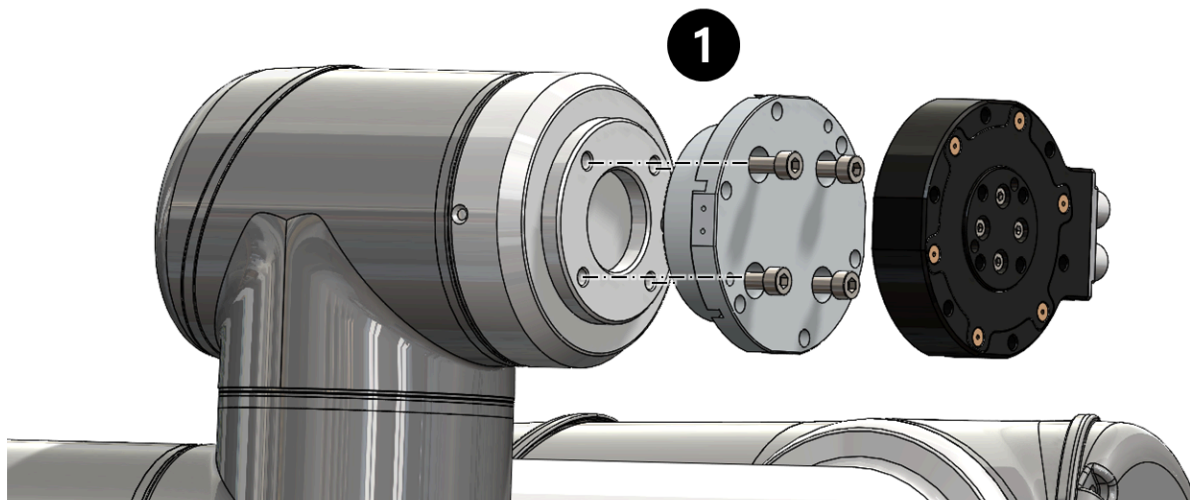
2.2 마운팅

- M5 볼트의 체결 토크는 5.2Nm입니다.
- 내부/외부 볼트를 분해할 경우 성능 보증이 불가능하며, A/S가 제공되지 않습니다.



*Transparent parts are user part examples

- 센서 체결순서



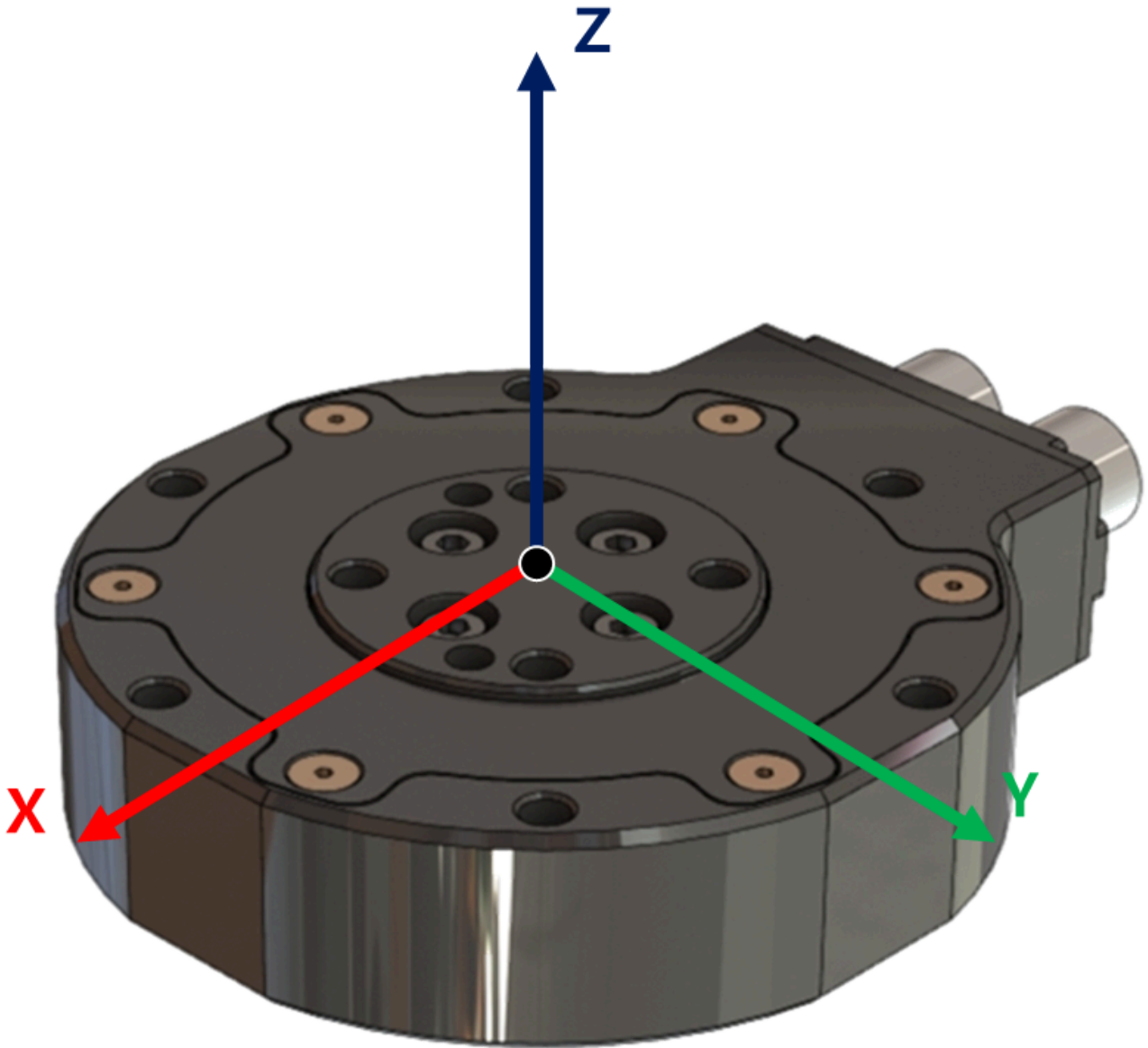
- 케이블 절단 및 과도한 당김에 대한 주의사항
- 센서 라인이 로봇 움직임에 따라 당겨지지 않도록 로봇과 함께 안전하게 고정해 주세요.
 - 케이블 타이 등으로 로봇에 직접 고정하거나, 다른 선들과 함께 묶어서 케이블 타이로 고정하는 것은 금지합니다.
 - 로봇과 고정할 때는 벨크로(찍찍이)를 사용하는 것을 권장하며, 다른 선들과 묶을 경우에도 벨크로를 사용해 고정해 주세요.

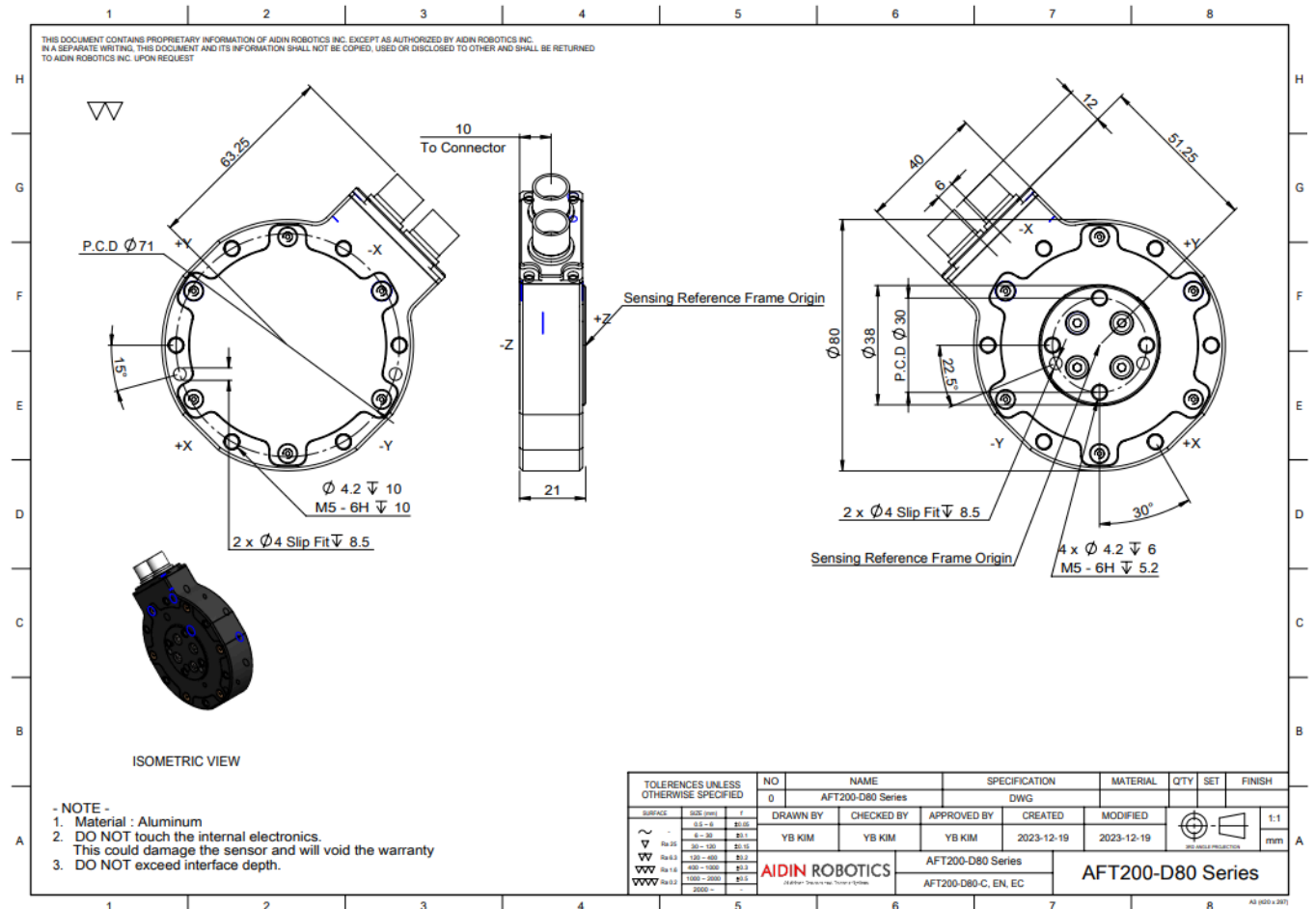
💡 체결 토크 가이드에 따르지 않을 경우 성능 보증이 불가능합니다.

💡 센서 라인이 로봇에 고정되지 않으면, 당기는 힘으로 인해 출력 신호에 노이즈가 발생할 수 있습니다.

💡 자세한 설치 가이드는 다음을 참고해 주세요: Smart 6-axis Force Torque Sensor, AFT200-D80, SETUP-GUIDE

2.3 축 & 도면





Drawing File :

- [AFT200_D80.pdf](#)

STEP File:

- [AFT200-D80-EC_배포용.zip](#)

💡 Port Location Notice

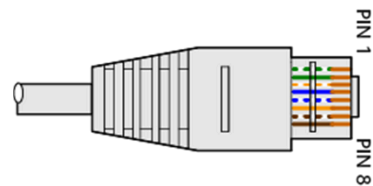
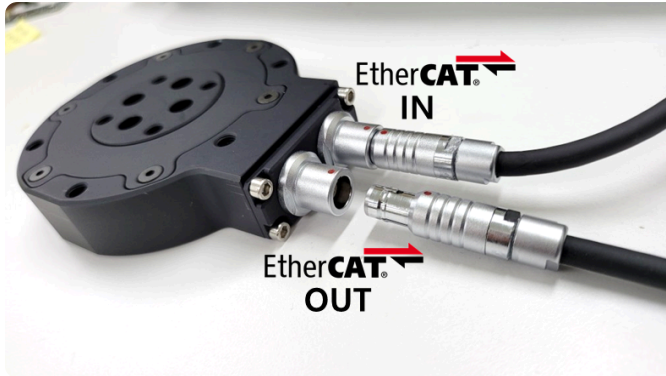
- **EtherCAT** Type model has two ports: **IN** and **OUT**.
- **Ethernet** Type model has one port, located on the **left side**.
- **CAN** Type model has one port, located on the **right side**.

2.4 와이어링

이더넷(EtherCAT) 장치는 POE 인젝터와 12-24V AC/DC 전원 공급장치와 함께 제공됩니다. 센서는 802.3af Mode A 프로토콜을 준수하지만, 전원은 Cat5e 이더넷 케이블 내 사용하지 않은 선을 통해서만 공급됩니다. 센서

케이블에 전압을 과도하게 주지 마십시오. 사용자는 센서 케이블을 인젝터의 'POE' 표시가 있는 포트에 연결해야 합니다. 802.3af Mode A와 호환되며 12-24V 전원 공급이 가능한 다른 POE 인젝터도 사용할 수 있습니다.

2.4.1 핀 홀



PIN	Wire	definition
PIN1	Green/White	Tx+
PIN2	Green	Tx-
PIN3	Orange/White	Rx+
PIN4	Blue	Vss+ (12-24V)
PIN5	Blue/White	
PIN6	Orange	Rx-
PIN7	Brown/White	
PIN8	Brown	Vss- (0V)



	Cable wires color	EtherCAT Signal
1	Green/White	TX+
2	Green	TX-
3	Orange/White	RX+
4	Blue	12V
5	Blue/White	NC
6	Orange	RX-

	Cable wires color	EtherCAT Signal
7	Brown/White	NC
8	Brown	GND

2.5 EtherCAT 인터페이스

AFT200-D80-EC 모델은 힘/토크 측정값을 제공하는 **RUN 상태** 를 지원합니다.

또한 AFT200-D80-EC 센서에 활성화 신호를 입력하면 IMU 측정 기능을 활성화할 수 있습니다.

IMU 신호를 활성화하거나 센서 신호의 바이어스를 설정하려면 2.6의 지침을 따르십시오.

EtherCAT 인터페이스를 통해 사용자는 다음과 같은 기능을 사용할 수 있습니다.

- 제품 번호, 시리얼 번호 등 읽기

- 힘/토크 데이터 읽기

→ 힘 [N] = Force output / 100-300, Torque = Torque output/500-50

- IMU 데이터 읽기

→ 가속도 [m/s²] = (Acceleration value - 32767) / 1000

→ 자이로스코프 [°/s] = (Gyroscope value - 32767) / 16.4

2.5.1 PDO 인터페이스(PDO interface)

PDO 인터페이스는 F/T 센서와 실시간으로 데이터를 교환합니다.

사용되는 PDO 매핑은 다음과 같습니다.

2.5.1.1 RxPDO1: Object 0x1600 디지털 출력

Name Type	Index	Subindex	Size	Bit Size	Access
Digital Output	0x7000	0x00	BOOL	1	RO

Name Type	Index	Subindex	Size	Bit Size	Access
BIAS_ON	0x7000	0x01	BOOL	1	RO
ACC_ON	0x7000	0x02	BOOL	1	RO
GYRO_ON	0x7000	0x03	BOOL	1	RO
NC	0x7000	0x04	BOOL	1	RO

2.5.1.2 TxPDO1: Object 0x1A00 디지털 출력

Name Type	Index	Subindex	Size	Bit Size	Access
Status	0x6000	0x00	SINT	8	RO
Force x	0x6000	0x01	REAL	32	RO
Force y	0x6000	0x02	REAL	32	RO
Force z	0x6000	0x03	REAL	32	RO
Torque x	0x6000	0x04	REAL	32	RO
Torque y	0x6000	0x05	REAL	32	RO
Torque z	0x6000	0x06	REAL	32	RO
Out of Range	0x6000	0x07	REAL	32	RO
Out of Range status	0x6000	0x08	REAL	32	RO
Acceleration x	0x6000	0x09	REAL	32	RO
Acceleration y	0x6000	0x0a	REAL	32	RO

Name Type	Index	Subindex	Size	Bit Size	Access
Acceleration z	0x6000	0x0b	REAL	32	RO
Angular Rate x	0x6000	0x0c	REAL	32	RO
Angular Rate y	0x6000	0x0d	REAL	32	RO
Angular Rate z	0x6000	0x0e	REAL	32	RO
Temperature	0x6000	0x0f	REAL	32	RO

2.5.2 SDO 인터페이스(SDO interface)

SDO 데이터는 센서를 구성하고, 제조 및 보정 데이터를 읽는 데 사용됩니다. 이 섹션에서는 EtherCAT F/T 센서 응용에 특화된 **사전(dictionary) 객체** 를 나열하며, EtherCAT 표준의 필수 객체는 포함하지 않습니다. 이 사전 객체들은 웹사이트에서 제공되는 **ESI 파일** 에서도 확인할 수 있습니다.

Object		Name	Type	Description
0x2000	0	Status	RO	
0x6000	0	Sensor Data	RO	Used to read the Force/Torque data, IMU etc. from the sensor
0x7000	0	Digital Output	RO	



센서 신호를 안정적으로 유지하기 위해 **약 30분간 작동** 하는 것을 권장합니다.

- 사용 전 **최소 10분 이상** 센서 데이터를 켜주시기 바랍니다.



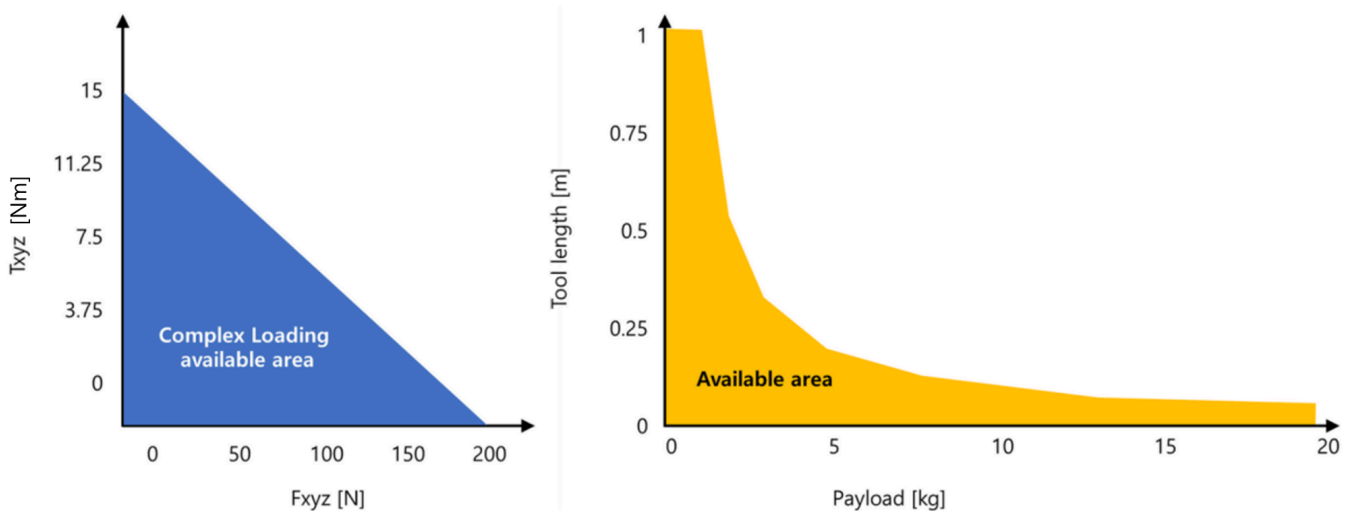
데이터를 켜 직후 10분간의 센서 출력 데이터는 다소 변동이 있을 수 있습니다.

2.6 힘/토크 범위 초과

- 센서는 단일 축 부하에서 정격 용량(nominal capacity)까지 동작할 수 있습니다.

정격 용량을 초과한 측정값은 올바르지 않으며 유효하지 않습니다.

- 복합 부하(composite loading)의 정격 용량은 아래 그림과 같은 시나리오로 나타낼 수 있습니다.
- 센서는 정상 동작 영역(normal operating area) 밖에서는 동작할 수 없습니다.



다음 그래프는 **높은 정밀도** 또는 **중간 정밀도** 가 요구되는 응용에서 AFT200-D80-EC 센서와 함께 사용할 수 있는 페이로드(payload) 및 툴(tool) 길이 범위를 보여줍니다.

F/T 부하는 센서의 측정 범위를 초과할 수 있습니다. TxPDO1[0x6000, Index: 0x08]은 값 타입(value type)을 나타내며, TxPDO1[0x6000, Index: 0x09]은 상태(status)를 표시합니다.

- Fxy와 Tz 축이 사용하는 보정 범위의 총 백분율이 105%를 초과하는 경우, 아래 Fxy, Tz 공식을 참조하십시오.

$$\sqrt{F_x^2 + F_y^2} / F_{xy}Range + |T_z| / T_zRange > 105\%$$

- Fz와 Txy 축이 사용하는 보정 범위의 총 백분율이 105%를 초과하는 경우, 아래 Fz, Txy 공식을 참조하십시오.

$$\sqrt{T_x^2 + T_y^2} / T_{xy}Range + |F_z| / F_zRange > 105\%$$

3. 소프트웨어(SOFTWARE)

3.1 Windows 애플리케이션에서 EtherCAT 테스트(TwinCAT3)

-  EtherCAT_manual.pdf

3.1.1 EtherCAT용 ESI(EtherCAT Slave Information) XML 파일

- XML 파일 이름:
 -  AFT200_D80_EC.xml
- XML 파일을 다운로드한 후 TwinCAT 설치 디렉터리에 붙여넣습니다.

경로 예: "C:/TwinCAT/3.1/Config/IO/EtherCAT"

3.1.2 테스트 순서(Sequence of Testing)

1. Visual Studio를 사용하여 TwinCAT 프로젝트를 생성합니다.
2. LAN 케이블을 EtherCAT 마스터(Ethernet 어댑터)에 연결합니다.
3. AFT 센서를 컵니다.
4. 장치를 스캔합니다.
5. TwinCAT을 프리런 모드(**free-run mode**) 로 활성화합니다.
6. **PDO(Process Data Object)** 및 **SDO(Service Data Object)** 의 출력 값을 관찰합니다.

3.2 Linux 애플리케이션에서 EtherCAT 테스트(SOEM)

3.2.1 SOEM 소스 코드 복제(Clone)

- 다음 페이지에서 SOEM 소스 코드를 복제합니다.

```
git clone https://github.com/OpenEtherCATsociety/SOEM
```

-  GitHub - OpenEtherCATsociety/SOEM: Simple Open Source EtherCAT Master

1. SOEM 웹페이지에 안내된 설치 단계를 따라 진행합니다.

```
mkdir build  
cd build  
cmake ..  
make
```

2. 다음 코드를 복사하여 **simpletest.c** 코드에 붙여넣습니다.

```

/** \file
 * \brief Example code for Simple Open EtherCAT master
 *
 * Usage : simple_test [ifname1]
 * ifname is NIC interface, f.e. eth0
 *
 * This is a minimal test.
 *
 * (c)Arthur Ketels 2010 - 2011
 */

#include <stdio.h>
#include <string.h>
#include <inttypes.h>

#include "ethercat.h"

#define EC_TIMEOUTMON 500

char IOmap[4096];
OSAL_THREAD_HANDLE thread1;
int expectedWKC;
boolean needlf;
volatile int wkc;
boolean inOP;
uint8 currentgroup = 0;
boolean forceByteAlignment = FALSE;

void simpletest(char *ifname)
{
    int i, j, oloop, iloop, chk;
    needlf = FALSE;
    inOP = FALSE;

    printf("Starting simple test\n");

    /* initialise SOEM, bind socket to ifname */
    if (ec_init(ifname))
    {
        printf("ec_init on %s succeeded.\n",ifname);
        /* find and auto-config slaves */

        if ( ec_config_init(FALSE) > 0 )
        {
            printf("%d slaves found and configured.\n",ec_slavecount);

            if (forceByteAlignment)
            {
                ec_config_map_aligned(&IOmap);
            }
        }
    }
}

```

```

    }
    else
    {
        ec_config_map(&IOmap);
    }

    ec_configdc();

    printf("Slaves mapped, state to SAFE_OP.\n");
    /* wait for all slaves to reach SAFE_OP state */
    ec_statecheck(0, EC_STATE_SAFE_OP, EC_TIMEOUTSTATE * 4);

    oloop = ec_slave[0].Obytes;
    if ((oloop == 0) && (ec_slave[0].Obits > 0)) oloop = 1;
    if (oloop > 8) oloop = 8;
    iloop = ec_slave[0].Ibytes;
    if ((iloop == 0) && (ec_slave[0].Ibits > 0)) iloop = 1;
    if (iloop > 8) iloop = 8;

    printf("segments : %d : %d %d %d %d\n", ec_group[0].nsegments, ec_group[0].IOsegment[0], ec_group[0].IOsegment[1], ec_group[0].IOsegment[2], ec_group[0].IOsegment[3]);

    printf("Request operational state for all slaves\n");
    expectedWKC = (ec_group[0].outputsWKC * 2) + ec_group[0].inputsWKC;
    printf("Calculated workcounter %d\n", expectedWKC);
    ec_slave[0].state = EC_STATE_OPERATIONAL;
    /* send one valid process data to make outputs in slaves happy*/
    ec_send_processdata();
    ec_receive_processdata(EC_TIMEOUTRET);
    /* request OP state for all slaves */
    ec_writestate(0);
    chk = 200;
    /* wait for all slaves to reach OP state */
    do
    {
        ec_send_processdata();
        ec_receive_processdata(EC_TIMEOUTRET);
        ec_statecheck(0, EC_STATE_OPERATIONAL, 50000);
    }
    while (chk-- && (ec_slave[0].state != EC_STATE_OPERATIONAL));
    if (ec_slave[0].state == EC_STATE_OPERATIONAL )
    {
        printf("Operational state reached for all slaves.\n");
        inOP = TRUE;
        /* cyclic loop */
        for(i = 1; i <= 10000; i++)
        {
            ec_send_processdata();
            wkc = ec_receive_processdata(EC_TIMEOUTRET);

```

```

        if(wkc >= expectedWKC)
        {
            printf("Processdata cycle %4d, WKC %d , 0:", i, wkc);

            for(j = 0 ; j < oloop; j++)
            {
                printf(" %2.2x", *(ec_slave[0].outputs + j));
            }

            printf(" I:");
            //Test
            int Fx= *(ec_slave[0].inputs+1)*256+*(ec_slave[0].inp
ts);
            int Fy= *(ec_slave[0].inputs+5)*256+*(ec_slave[0].inp
ts+4);
            int Fz= *(ec_slave[0].inputs+9)*256+*(ec_slave[0].inp
ts+8);
            int Tx= *(ec_slave[0].inputs+13)*256+*(ec_slave[0].inp
uts+12);
            int Ty= *(ec_slave[0].inputs+17)*256+*(ec_slave[0].inp
uts+16);
            int Tz= *(ec_slave[0].inputs+21)*256+*(ec_slave[0].inp
uts+20);
            int Ax= *(ec_slave[0].inputs+25)*256+*(ec_slave[0].inp
uts+24);
            int Ay= *(ec_slave[0].inputs+29)*256+*(ec_slave[0].inp
uts+28);
            int Az= *(ec_slave[0].inputs+33)*256+*(ec_slave[0].inp
uts+32);

            printf(" Fx: %d, Fy: %d,Fz: %d,Tx: %d,Ty: %d,Tz: %d,A
x: %d,Ay: %d,Az: %d, ", Fx, Fy,Fz,Tx,Ty,Tz,Ax,Ay,Az);
            printf(" T:%"PRIu64"\n",ec_DCtime);
            needlf = TRUE;
        }
        osal_usleep(5000);
    }
    inOP = FALSE;
}
else
{
    printf("Not all slaves reached operational state.\n");
    ec_readstate();
    for(i = 1; i<=ec_slavecount ; i++)
    {
        if(ec_slave[i].state != EC_STATE_OPERATIONAL)
        {
            printf("Slave %d State=0x%2.2x StatusCode=0x%4.4x : %s

```

```

\n",
                                i, ec_slave[i].state, ec_slave[i].ALstatuscode, ec
_ALstatuscode2string(ec_slave[i].ALstatuscode));
        }
    }
    }
    printf("\nRequest init state for all slaves\n");
    ec_slave[0].state = EC_STATE_INIT;
    /* request INIT state for all slaves */
    ec_writestate(0);
}
else
{
    printf("No slaves found!\n");
}
printf("End simple test, close socket\n");
/* stop SOEM, close socket */
ec_close();
}
else
{
    printf("No socket connection on %s\nExecute as root\n",ifname);
}
}

OSAL_THREAD_FUNC ecatcheck( void *ptr )
{
    int slave;
    (void)ptr;                /* Not used */

    while(1)
    {
        if( inOP && ((wkc < expectedWKC) || ec_group[currentgroup].docheckstat
e))
        {
            if (needlf)
            {
                needlf = FALSE;
                printf("\n");
            }
            /* one ore more slaves are not responding */
            ec_group[currentgroup].docheckstate = FALSE;
            ec_readstate();
            for (slave = 1; slave <= ec_slavecount; slave++)
            {
                if ((ec_slave[slave].group == currentgroup) && (ec_slave[slav
e].state != EC_STATE_OPERATIONAL))
                {
                    ec_group[currentgroup].docheckstate = TRUE;
                    if (ec_slave[slave].state == (EC_STATE_SAFE_OP + EC_STATE_ER

```

```

ROR) )
    {
        printf("ERROR : slave %d is in SAFE_OP + ERROR, attemptin
g ack.\n", slave);
        ec_slave[slave].state = (EC_STATE_SAFE_OP + EC_STATE_AC
K);
        ec_writestate(slave);
    }
    else if(ec_slave[slave].state == EC_STATE_SAFE_OP)
    {
        printf("WARNING : slave %d is in SAFE_OP, change to OPERA
TIONAL.\n", slave);
        ec_slave[slave].state = EC_STATE_OPERATIONAL;
        ec_writestate(slave);
    }
    else if(ec_slave[slave].state > EC_STATE_NONE)
    {
        if (ec_reconfig_slave(slave, EC_TIMEOUTMON))
        {
            ec_slave[slave].islost = FALSE;
            printf("MESSAGE : slave %d reconfigured\n",slave);
        }
    }
    else if(!ec_slave[slave].islost)
    {
        /* re-check state */
        ec_statecheck(slave, EC_STATE_OPERATIONAL, EC_TIMEOUTRE
T);

        if (ec_slave[slave].state == EC_STATE_NONE)
        {
            ec_slave[slave].islost = TRUE;
            printf("ERROR : slave %d lost\n",slave);
        }
    }
}
if (ec_slave[slave].islost)
{
    if(ec_slave[slave].state == EC_STATE_NONE)
    {
        if (ec_recover_slave(slave, EC_TIMEOUTMON))
        {
            ec_slave[slave].islost = FALSE;
            printf("MESSAGE : slave %d recovered\n",slave);
        }
    }
    else
    {
        ec_slave[slave].islost = FALSE;
        printf("MESSAGE : slave %d found\n",slave);
    }
}

```

```

        }
    }
    if(!ec_group[currentgroup].docheckstate)
        printf("OK : all slaves resumed OPERATIONAL.\n");
    }
    osal_usleep(10000);
}
}

int main(int argc, char *argv[])
{
    printf("SOEM (Simple Open EtherCAT Master)\nSimple test\n");

    if (argc > 1)
    {
        /* create thread to handle slave error handling in OP */
        // pthread_create( &thread1, NULL, (void *) &ecatcheck, (void*) &ctime);
        osal_thread_create(&thread1, 128000, &ecatcheck, (void*) &ctime);
        /* start cyclic part */
        simpletest(argv[1]);
    }
    else
    {
        ec_adaptert * adapter = NULL;
        printf("Usage: simple_test ifname\nifname = eth0 for example\n");

        printf ("\nAvailable adapters:\n");
        adapter = ec_find_adapters ();
        while (adapter != NULL)
        {
            printf ("    - %s (%s)\n", adapter->name, adapter->desc);
            adapter = adapter->next;
        }
        ec_free_adapters(adapter);
    }

    printf("End program\n");
    return (0);
}

```

3. **build** 폴더에서 make 명령을 실행합니다.
4. 이후 다음과 같이 **simpletest** 프로그램을 실행합니다.

```

SOEM/build/test/linux/simple_test
sudo ./simple_test -(port)

```

```

gene@gene-GENE-KBU6: ~/Downloads/SOEM/build/test/linux/simple_test
Processdata cycle 9962, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16122, Fz: 16174, Tx: 16294, Ty: 16313, Tz: 15962, Ax: 15971, Ay: 15948, Az: 20714, T:729257737049616641
Processdata cycle 9963, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16121, Fz: 16175, Tx: 16294, Ty: 16314, Tz: 15970, Ax: 15971, Ay: 15947, Az: 20713, T:729257737054901961
Processdata cycle 9964, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16120, Fz: 16177, Tx: 16295, Ty: 16317, Tz: 15966, Ax: 15970, Ay: 15948, Az: 20713, T:729257737060188681
Processdata cycle 9965, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16177, Tx: 16294, Ty: 16314, Tz: 15963, Ax: 15971, Ay: 15950, Az: 20713, T:729257737065473520
Processdata cycle 9966, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16121, Fz: 16175, Tx: 16295, Ty: 16313, Tz: 15969, Ax: 15971, Ay: 15947, Az: 20713, T:729257737070759280
Processdata cycle 9967, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16174, Tx: 16294, Ty: 16314, Tz: 15965, Ax: 15971, Ay: 15946, Az: 20710, T:729257737076046040
Processdata cycle 9968, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16120, Fz: 16175, Tx: 16294, Ty: 16315, Tz: 15965, Ax: 15970, Ay: 15943, Az: 20715, T:729257737081332560
Processdata cycle 9969, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16176, Tx: 16294, Ty: 16317, Tz: 15960, Ax: 15971, Ay: 15944, Az: 20712, T:729257737086623440
Processdata cycle 9970, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16121, Fz: 16176, Tx: 16295, Ty: 16314, Tz: 15968, Ax: 15970, Ay: 15946, Az: 20713, T:729257737091911640
Processdata cycle 9971, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16176, Tx: 16294, Ty: 16312, Tz: 15969, Ax: 15970, Ay: 15947, Az: 20715, T:729257737097197720
Processdata cycle 9972, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16176, Tx: 16295, Ty: 16315, Tz: 15964, Ax: 15970, Ay: 15947, Az: 20713, T:729257737102482641
Processdata cycle 9973, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16120, Fz: 16176, Tx: 16295, Ty: 16315, Tz: 15961, Ax: 15970, Ay: 15948, Az: 20712, T:729257737107767001
Processdata cycle 9974, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16118, Fz: 16176, Tx: 16294, Ty: 16315, Tz: 15961, Ax: 15971, Ay: 15946, Az: 20715, T:729257737113052361
Processdata cycle 9975, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16176, Tx: 16295, Ty: 16316, Tz: 15963, Ax: 15970, Ay: 15946, Az: 20714, T:729257737118337920
Processdata cycle 9976, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16118, Fz: 16178, Tx: 16295, Ty: 16314, Tz: 15970, Ax: 15971, Ay: 15948, Az: 20714, T:729257737123622960
Processdata cycle 9977, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16121, Fz: 16176, Tx: 16294, Ty: 16311, Tz: 15961, Ax: 15972, Ay: 15950, Az: 20714, T:729257737128908360
Processdata cycle 9978, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16118, Fz: 16176, Tx: 16294, Ty: 16315, Tz: 15965, Ax: 15971, Ay: 15945, Az: 20712, T:729257737134193680
Processdata cycle 9979, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16120, Fz: 16175, Tx: 16295, Ty: 16316, Tz: 15967, Ax: 15971, Ay: 15946, Az: 20711, T:729257737139479040
Processdata cycle 9980, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16122, Fz: 16175, Tx: 16296, Ty: 16315, Tz: 15966, Ax: 15970, Ay: 15947, Az: 20713, T:729257737144765240
Processdata cycle 9981, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16118, Fz: 16175, Tx: 16292, Ty: 16315, Tz: 15965, Ax: 15971, Ay: 15945, Az: 20715, T:729257737150050920
Processdata cycle 9982, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16120, Fz: 16175, Tx: 16296, Ty: 16315, Tz: 15967, Ax: 15970, Ay: 15946, Az: 20715, T:729257737155337120
Processdata cycle 9983, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16175, Tx: 16295, Ty: 16314, Tz: 15968, Ax: 15970, Ay: 15948, Az: 20711, T:729257737160625480
Processdata cycle 9984, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16176, Tx: 16294, Ty: 16312, Tz: 15965, Ax: 15969, Ay: 15946, Az: 20713, T:729257737165907160
Processdata cycle 9985, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16177, Tx: 16294, Ty: 16315, Tz: 15967, Ax: 15971, Ay: 15946, Az: 20713, T:729257737171202400
Processdata cycle 9986, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16121, Fz: 16176, Tx: 16294, Ty: 16315, Tz: 15967, Ax: 15970, Ay: 15946, Az: 20712, T:729257737176481920
Processdata cycle 9987, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16117, Fz: 16175, Tx: 16294, Ty: 16311, Tz: 15965, Ax: 15971, Ay: 15948, Az: 20715, T:729257737181763400
Processdata cycle 9988, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16177, Tx: 16295, Ty: 16316, Tz: 15963, Ax: 15970, Ay: 15945, Az: 20711, T:729257737187049160
Processdata cycle 9989, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16117, Fz: 16177, Tx: 16295, Ty: 16315, Tz: 15958, Ax: 15971, Ay: 15946, Az: 20712, T:729257737192340960
Processdata cycle 9990, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16120, Fz: 16175, Tx: 16293, Ty: 16315, Tz: 15962, Ax: 15971, Ay: 15946, Az: 20714, T:729257737197624360
Processdata cycle 9991, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16176, Tx: 16294, Ty: 16317, Tz: 15963, Ax: 15970, Ay: 15947, Az: 20713, T:729257737202910600
Processdata cycle 9992, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16175, Tx: 16294, Ty: 16313, Tz: 15964, Ax: 15971, Ay: 15946, Az: 20714, T:729257737208205520
Processdata cycle 9993, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16121, Fz: 16177, Tx: 16294, Ty: 16315, Tz: 15962, Ax: 15971, Ay: 15945, Az: 20714, T:729257737213490720
Processdata cycle 9994, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16120, Fz: 16177, Tx: 16294, Ty: 16316, Tz: 15962, Ax: 15972, Ay: 15946, Az: 20711, T:729257737218775640
Processdata cycle 9995, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16120, Fz: 16177, Tx: 16295, Ty: 16314, Tz: 15968, Ax: 15971, Ay: 15947, Az: 20712, T:729257737224060640
Processdata cycle 9996, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16118, Fz: 16177, Tx: 16293, Ty: 16314, Tz: 15966, Ax: 15970, Ay: 15947, Az: 20715, T:729257737229346160
Processdata cycle 9997, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16121, Fz: 16177, Tx: 16294, Ty: 16314, Tz: 15963, Ax: 15970, Ay: 15946, Az: 20715, T:729257737234631560
Processdata cycle 9998, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16119, Fz: 16175, Tx: 16294, Ty: 16315, Tz: 15965, Ax: 15971, Ay: 15947, Az: 20711, T:729257737239916520
Processdata cycle 9999, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16120, Fz: 16176, Tx: 16294, Ty: 16316, Tz: 15967, Ax: 15970, Ay: 15946, Az: 20712, T:729257737245212561
Processdata cycle 10000, WKC 3 , 0: 00 I: Fx: 12345, Fy: 16117, Fz: 16176, Tx: 16295, Ty: 16313, Tz: 15965, Ax: 15970, Ay: 15945, Az: 20715, T:729257737250497320

```

Revision History

날짜	내용
----	----

2026.09.15

초기 버전 (현재 버전)



찾으시는 내용이 없나요?

자주 묻는 질문을 먼저 확인해 보세요.

해결되지 않으면 문의하기를 작성해 주세요.

자주 묻는 질문

문의하기 →