Nuvoton Fly-Controller Development Kit

Product No.	**M4 Fly-Controller Application**
Function	**Development Kit and Library**

File Information	
Name	**Nuvoton Fly-Controller Development Kit.pdf**
Project	**M4 Application**
Function	**Controller and Library Description**
Purpose	
Author	**TZU-LAN SHEN (tlshen)**

Revision History

Revision	Date	Comments
1.0	15/2/2	Original Plan

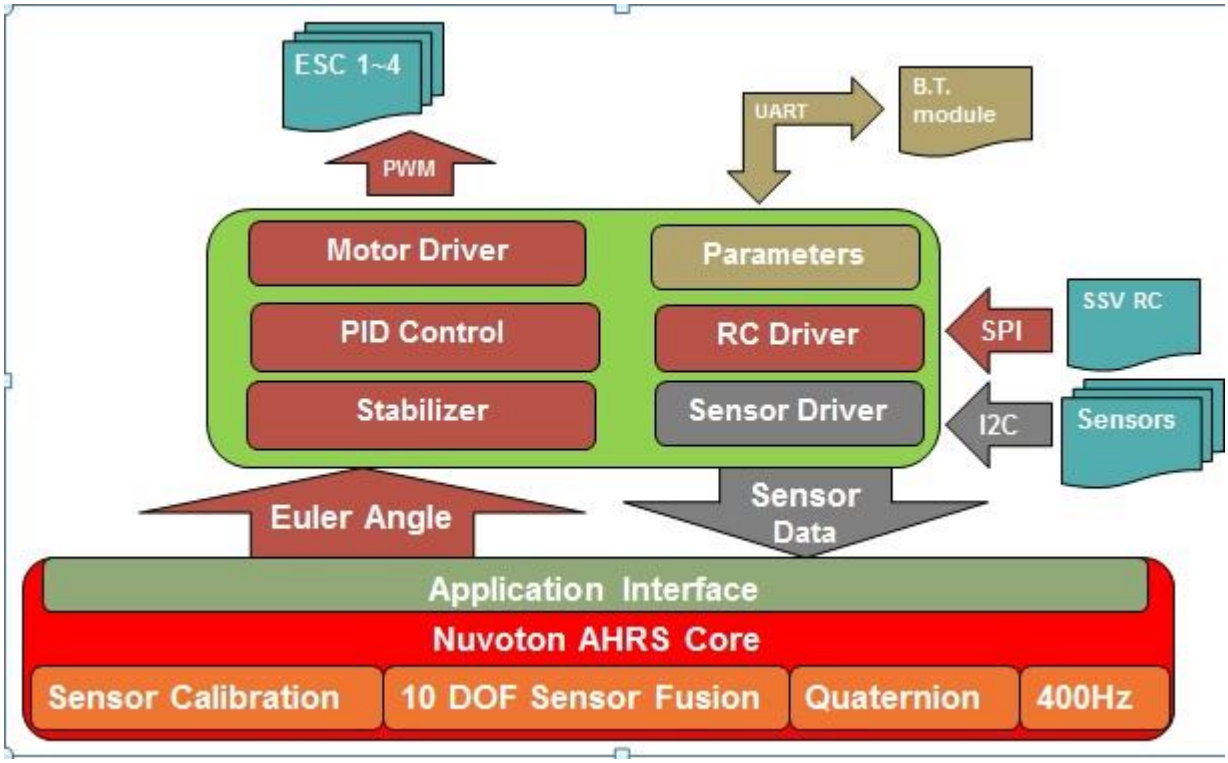
Release Note:

Revision 1.0 **(1st Draft Version)**

1. Overview of Fly-Controller Development Environment

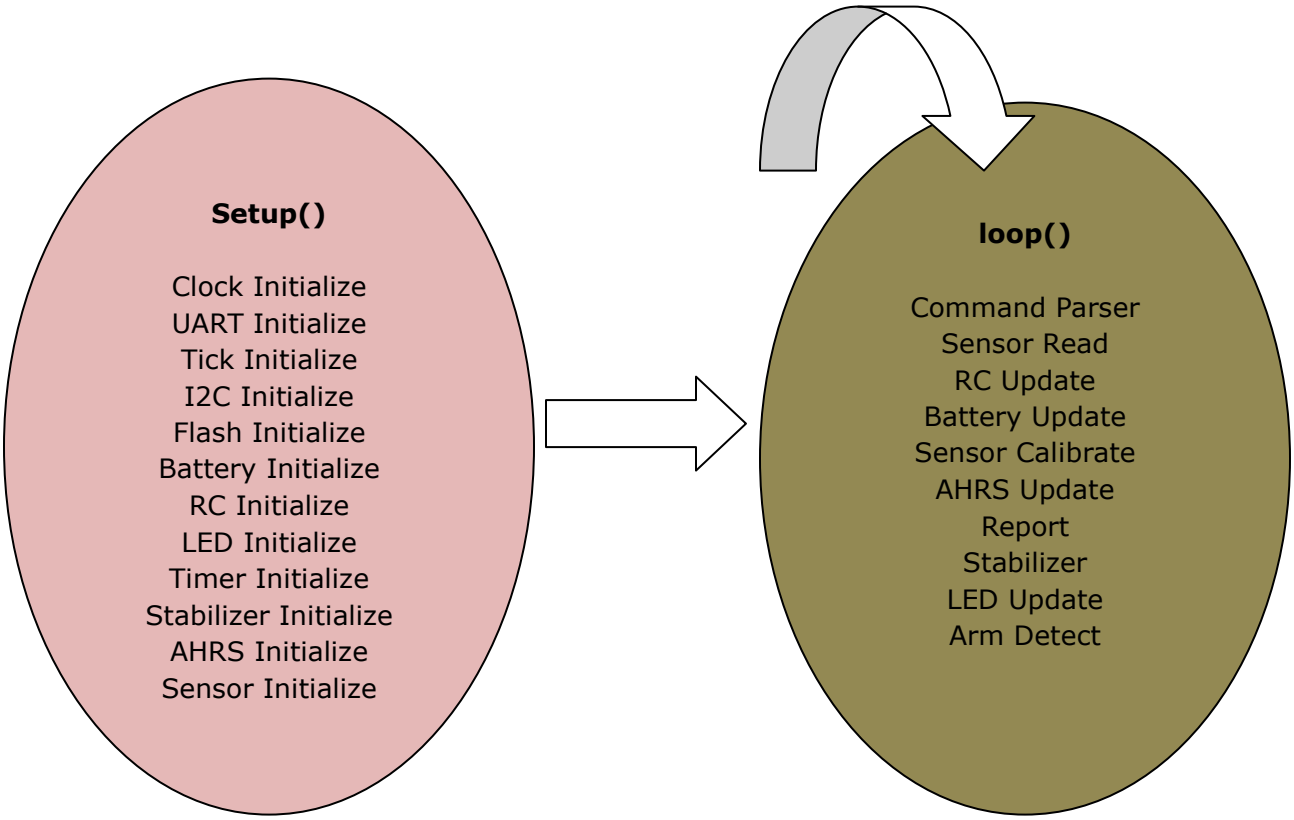
- Firmware Development Environment
  - ARM MDK uVision V4.60
  - Nuvoton Nu-Link ice adapter
  - SDK Package NvtFly.7z
  - FCA Project - NvtFly\M451\NvyFly\FlyController\FUSIONSDK
  - AHRS header - NvtFly\M451\NvyFly\FlyController\AHRSLib\include
  - AHRS library - NvtFly\M451\NvyFly\FlyController\AHRSLib\obj

- Firmware Architecture
  - Fly Controller Application (FCA), which provide customers a reference fly control frame work of Nuvoton Cortex-M4 and allow user to modify for their own sensors, radio controls, motors, stabilizer and any other possible extensions.
  - Attitude and head Reference System library (AHRS) provides the application interfaces for users to process sensor raw data, sensor calibrations and the Euler angle of sensor fusion.


ESC 1~4
PWM
UART
B.T. module
Motor Driver
Parameters
PID Control
RC Driver
Stabilizer
Sensor Driver
SPI
SSV RC
I2C
Sensors
Euler Angle
Sensor Data
Application Interface
Nuvoton AHRS Core
Sensor Calibration
10 DOF Sensor Fusion
Quaternion
400Hz

2. FCA Operations

- **System Initialization – Setup()**
  - Setup is an initialization process of platform driver, fly-controller and AHRS library.
  - Setup must be called before any applications and user can extend any self-defined initialization process here.
- **Main Control Routine – loop()**
  - This is an infinite routine to handle the main control process of fly controller.
  - CommandProcess() – handle the console command of the fly controller.
  - SensorRead() – Read raw data from sensor
  - ssv_rc_update() – If use SSV RC, this process update RC control input from SSV Receiver.
  - UpdateBattery() – Read ADC to update battery capacity.
  - SensorDynamicCalibrate() – Do sensor calibration if needed.
  - computeRC() – Convert RC input raw data (SSV RC/PWM RC) to fly controller control format.
  - armDetect() – Process copter arm/dis-arm and copter Flip then dis-arm protection.
  - nvtUpdateAHRS(SENSOR_ACC|SENSOR_GYRO|SENSOR_MAG) – Compute attitude and heading information by AHRS lib.
  - report_sensors() – This process display the information about sensors, fly controllers, RC, motors depend on what console command you have triggered.
  - stabilizer() – The main function of fly-controller. It gets AHRS information from library and fusion with RC controls to come up with the final motor speed control.
  - UpdateLED() – Response the status and mode by on board LED.



FCA Operation Diagram

3. AHRS Operations

- **System Initialize Phase**
  - nvtAHRSInit() must be called before any AHRS operations.
  - nvtMilisecondTick() must be called every millisecond to provide a clock source for AHRS library.

- **Sensor Initialize Phase**
  - To do ACC initialize for AHRS, you need to setup ACC scale, offset and gravity scale by nvtSetAccScale(), nvtSetAccOffset() and nvtSetAccG_PER_LSB() before using ACC.
  - To do GYRO initialize for AHRS, you need to setup GYRO scale, offset and degree scale by nvtSetGyroScale(), nvtSetGyroOffset() and nvtSetGyroDegPLSB() before using GYRO .
  - To do MAG initialize for AHRS, you need to setup MAG calibration matrix and magnetic gauss scale by nvtSetMagCalMatrix() and nvtSetMagGaussPLSB() before using MAG .

- **Sensor Fusion Phase**
  - *Raw Data Input* - AHRS need applications to input sensor raw data before fusion process. You can call :
    - ✧ nvtInputSensorRawBARO() to input BARO raw data.
    - ✧ nvtInputSensorRawACC() to input ACC raw data.

- ✧ nvtInputSensorRawMAG() to input MAG raw data.
- ✧ nvtInputSensorRawGYRO() to input GYRO raw data.

- *AHRS Calculation* – when sensor raw data has been ready in AHRS, application can call nvtUpdateAHRS() to start AHRS calculation.

- *AHRS output* – after AHRS caculation, application can call nvtGetEulerRPY() to get the fusion result of Euler angle by Roll degree , Pitch degree and Yaw degree. The other AHRS output function nvtGetVelocity() provides velocity information for object motion which is useful in altitude hold control.

- **Sensor Calibration Phase**

AHRS can do sensor calibration for applications to get a more accurate attitude information, however this is not a "must" process for AHRS computation.

- ACC Calibration – There are 2 ways to do ACC calibration, one called "Fast Z" calibration and the other is "Full Face" calibration. Both method needs application to start with the nvtCalACCInit() function before ACC calibration. The "Full Face" calibration needs application to input all 6 faces (+X, -X, +Y, -Y, +Z, -Z) ACC data by earth gravity. Application repeats reading ACC raw data from sensor and input AHRS by function nvtCalACCBufferFill() which returns token "STATUS_BUFFER_FILLED" when one side is ready. Application can input 6 faces in any order and redo some faces until all 6 faces' probing complete. Once all faces complete, AHRS do the ACC calibration and output token "STATUS_CAL_DONE". Application can further get the calibration result by nvtGetAccOffset() and nvtGetAccScale(). "Fast Z" calibration

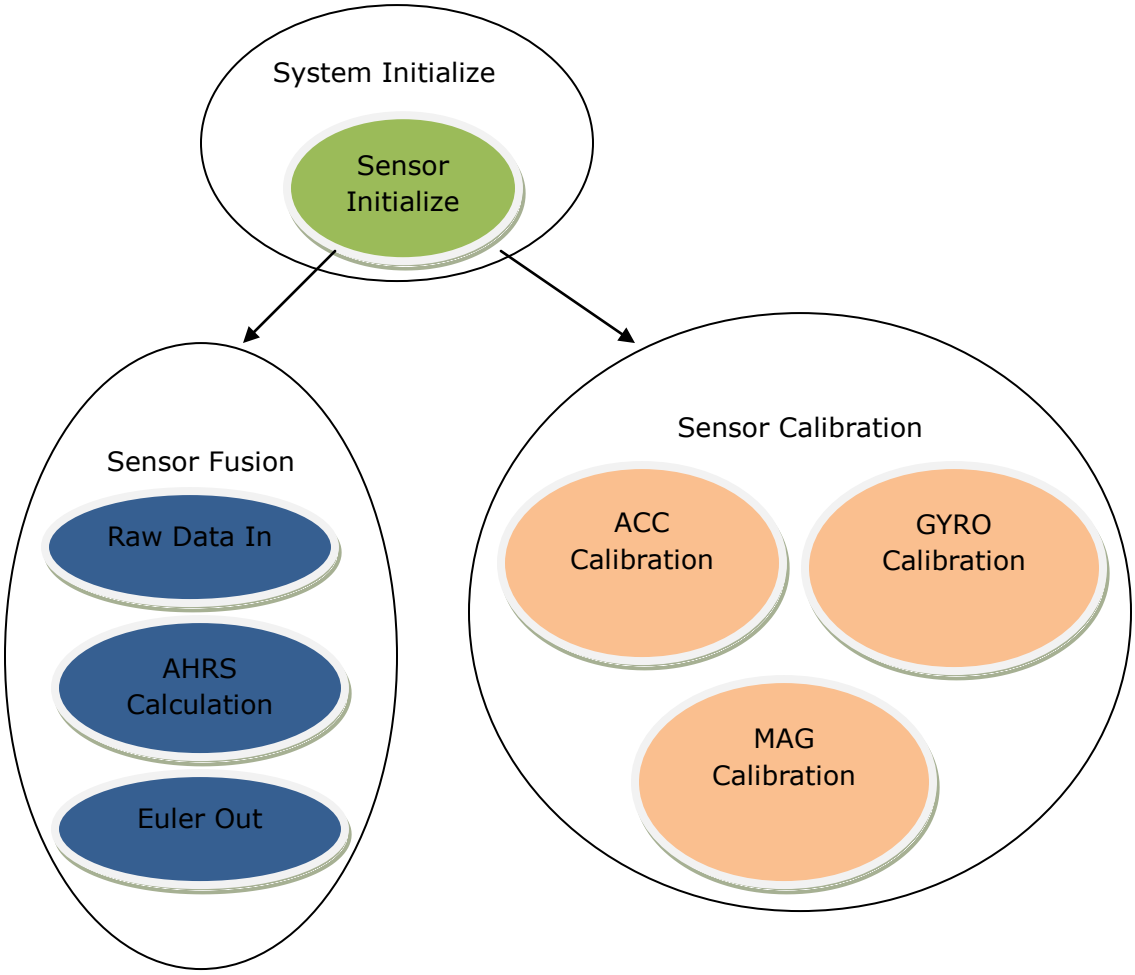
provide a fast and one size(Z) only calibration method. Only input sensor Z data to AHRS by side "0" and AHRS do ACC calibration by side Z('0') only.

- GYRO Calibration – There are 2 kinds of calibration applied for GYRO sensor, one called "Scale" calibration and the other called "Drift" calibration. Application needs to call nvtCalGyroInit() before doing GYRO scale calibration. "Scale" calibration compensates the degree error by multiply scale factors for axis X, Y and Z while GRYO rotation. For example, to do the GYRO Z scale calibration, you need to put sensor on a flat table with Z up, and repeat
  - ✧ probe GYRO Z data into AHRS
  - ✧ Call nvtGyroScaleCalibrate() and check return value

  Meanwhile, rotate sensor 3 turns (360 degreex3) around Z axis and keep sensor steady until token "STATUS_GYRO_CAL_DONE" return by nvtGyroScaleCalibrate(). Apply the same actions to calibrate axis X and Y.

  "Drift" calibration needs sensor to be steady, and meanwhile call nvtGyroCenterCalibrate() until token "STATUS_GYRO_CAL_DONE". Application can then call nvtGetCalibratedGYRO() to get the calibrated gyro.
- MAG Calibration – Always start with the nvtCalMAGInit() function before MAG calibration and repeat,
  - ✧ probe MAG data into AHRS
  - ✧ Call nvtCalMAGBufferFill() and check return value

  Meanwhile, rotate sensor around X, Y and Z axis for more than 360 degree until token "STATUS_CAL_DONE" return by nvtCalMAGBufferFill(). Apply nvtGetMagCalQFactor() to get a MAG calibration quality factor, 255(worst) to 0(best). Get a Q factor < 10 is good enough for common applications.



AHRS Operation Diagram

4. AHRS Library Interface

- **Headers and LIBs of AHRS**
  - "AHRSLib.h" – AHRS interface file header, FCA should include this header for a reference to AHRS library.
  - "AHRSLib.lib" – AHRS core library, FCA project should include this library for AHRS linking library.
- **Function Interfaces**

NAME	**nvtAHRSInit**	
SYNOPSIS	**void nvtAHRSInit(void)**	
Description	Do the state initialize of AHRS library. This function should be called before using AHRS library.	
Parameters	**None**	
Return Value	void	

NAME	**nvtMillisecondTick**	
SYNOPSIS	**void nvtMillisecondTick(void)**	
Description	AHRS needs applications call this function every millisecond to provide a millisecond tick for AHRS operaion.	
Parameters	**None**	
Return Value	void	

NAME	**nvtSetAccScale**

SYNOPSIS	**void nvtSetAccScale(float* AccScale);**	
Description	Set ACC scale factor to AHRS and compensate ACC scale error.	
Parameters	**Float* AccScale{3}**	ACC scale X,Y and Z
Return Value	void	

NAME	**nvtSetAccOffset**	
SYNOPSIS	**void nvtSetAccOffset(float* AccMean);**	
Description	Set ACC offset factor to AHRS and compensate ACC mean error.	
Parameters	**Float* AccMean{3}**	ACC mean X, Y, and Z
Return Value	void	

NAME	**nvtSetAccG_PER_LSB**	
SYNOPSIS	**void nvtSetAccG_PER_LSB(float G_PER_LSB);**	
Description	Tells the AHRS how raw data to be converted into a unified gravity (1G)	
Parameters	**Float G_PER_LSB**	A mapping factor from sensor raw to 1G, usually depend on sensor setting
Return Value	void	

NAME	**nvtSetGyroScale**	
SYNOPSIS	**void nvtSetGyroScale(float* GyroScale);**	
Description	Set GYRO scale factor to AHRS and compensate GYRO scale error.	
Parameters	**Float* GyroScale{3}**	GYRO scale X,Y and Z
Return Value	void	

NAME	**nvtSetGyroOffset**	
SYNOPSIS	**void nvtSetGyroOffset(float* GyroMean);**	
Description	Set GYRO offset factor to AHRS and compensate GYRO mean error.	
Parameters	**Float* GyroMean {3}**	GYRO mean X, Y, and Z
Return Value	void	

NAME	**nvtSetGYRODegPLSB**	
SYNOPSIS	**void nvtSetGYRODegPLSB(float DPLSB);**	
Description	Tells the AHRS how raw data to be converted into a unified degree (1 degree)	
Parameters	**Float DPLSB**	A mapping factor from sensor raw to one degree, usually depend on sensor settings
Return Value	void	

NAME	**nvtSetMagCalMatrix**	
SYNOPSIS	**void nvtSetMagCalMatrix(float* MagCalMatrix);**	
Description	Set MAG calibration matrix to AHRS and compensate MAG soft iron and hard iron.	
Parameters	**Float* MagCalMatrix {10}**	An array pointer to 10 float parameters for MAG gauss distortion fix.
Return Value	void	

NAME	**nvtSetMagGaussPLSB**	
SYNOPSIS	**void nvtSetMagGaussPLSB(float)**	
Description	Tells the AHRS how raw data to be converted into a unified gauss (1 gauss)	
Parameters	**Float**	A mapping factor from sensor raw to one gauss, usually depend on sensor setting
Return Value	void	

NAME	**nvtInputSensorRawBARO**	
SYNOPSIS	**void nvtInputSensorRawBARO(int16_t *raw)**	
Description	Input BARO raw data into AHRS	
Parameters	**Int16* raw{2}**	Raw[0]:Raw temperature from sensor Raw[1]:Raw pressure from sensor
Return Value	void	

NAME	**nvtInputSensorRawBARO**	
SYNOPSIS	**void nvtInputSensorRawBARO(int16_t *raw)**	
Description	Input BARO raw data into AHRS	
Parameters	**Int16* raw{2}**	Raw[0]:Raw temperature from sensor Raw[1]:Raw pressure from sensor
Return Value	void	

NAME	**nvtInputSensorRawACC**	
SYNOPSIS	**void nvtInputSensorRawACC(int16_t *raw)**	
Description	Input ACC raw data into AHRS	
Parameters	**Int16* raw{3}**	Raw[0]: ACC X Raw[1]: ACC Y Raw[1]: ACC Z
Return Value	void	

NAME	**nvtInputSensorRawGYRO**	
SYNOPSIS	**void nvtInputSensorRawGYRO(int16_t *raw);**	
Description	Input GYRO raw data into AHRS	
Parameters	**Int16* raw{3}**	Raw[0]: GYRO X Raw[1]: GYRO Y Raw[1]: GYRO Z
Return Value	void	

NAME	**nvtInputSensorRawMAG**	
SYNOPSIS	**void nvtInputSensorRawMAG(int16_t *raw);**	
Description	Input MAG raw data into AHRS	
Parameters	**Int16* raw{3}**	Raw[0]: MAG X Raw[1]: MAG Y Raw[1]: MAG Z
Return Value	void	

NAME	**nvtGetSensorRawBARO**	
SYNOPSIS	**void nvtGetSensorRawBARO(int16_t *raw)**	
Description	Get BARO raw data from AHRS	
Parameters	**int16* raw{2}**	Raw[0]:Raw temperature from sensor Raw[1]:Raw pressure from sensor
Return Value	void	

NAME	**nvtGetSensorRawACC**	
SYNOPSIS	**void nvtGetSensorRawACC(int16_t *raw)**	
Description	Get ACC raw data from AHRS	
Parameters	**int16* raw{3}**	Raw[0]: ACC X Raw[1]: ACC Y Raw[1]: ACC Z
Return Value	void	

NAME	**nvtGetSensorRawGYRO**	
SYNOPSIS	**void nvtGetSensorRawGYRO(int16_t *raw);**	
Description	Get GYRO raw data from AHRS	
Parameters	**int16* raw{3}**	Raw[0]: GYRO X Raw[1]: GYRO Y Raw[1]: GYRO Z
Return Value	void	

NAME	**nvtGetSensorRawMAG**	
SYNOPSIS	**void nvtGetSensorRawMAG(int16_t *raw);**	
Description	Get MAG raw data from AHRS	
Parameters	**int16* raw{3}**	Raw[0]: MAG X Raw[1]: MAG Y Raw[1]: MAG Z
Return Value	void	

NAME	**nvtUpdateAHRS**	
SYNOPSIS	**void nvtUpdateAHRS(uint8_t UPDATE);**	
Description	Start AHRS calculation and come up with Euler angle and Quaternion. Application can select sensors by active bits from input parameter 'UPDATE'. Sensors with bit '1' are going to be calculated by AHRS.	
Parameters	**uint8_t UPDATE**	Bit0: update with ACC Bit1: update with GYRO Bit2: update with MAG Bit3: update with BARO
Return Value	void	

NAME	**nvtGetEulerRPY**
SYNOPSIS	**void nvtGetEulerRPY(float*);**
Description	Get the Euler angle previously calculated by AHRS

Parameters	**float* {3}**	[0]: Roll angle in degree [1]: Pitch angle in degree [2]: Yaw angle in degree
Return Value	void	

NAME	**nvtGetVelocity**	
SYNOPSIS	**void nvtGetVelocity(float* Velocity);**	
Description	Get the estimated velocity previously calculated by AHRS. The parameter returns the velocity (meter/second) along sensor ACC axis X, Y and Z. This needs application to input sensor ACC raw and turn on bit "update with ACC".	
Parameters	**float* Velocity {3}**	Velocity [0]: X velocity. Velocity [1]: Y velocity. Velocity [2]: Z velocity.
Return Value	void	

NAME	**nvtCalACCInit**	
SYNOPSIS	**void nvtCalACCInit(void)**	
Description	Application should call this function before processing ACC calibration.	
Parameters	**None**	
Return Value	void	

NAME	**nvtCalACCBufferFill**

SYNOPSIS	**signed char nvtCalACCBufferFill(int8_t Dir)**	
Description	Input ACC raw data into AHRS buffer. For full calibration, application needs to set input parameter 'Dir' from 0 ~ 5, to sample all 6 faces ACC raw data into buffer.	
Parameters	**int8_t Dir**	0: Sample side 0 1: Sample side 1 2: Sample side 2 3: Sample side 3 4: Sample side 4 5: Sample side 5
Return Value	STATUS_BUFFER_FILLED : Side x buffer sample done STATUS_BUFFER_NOT_FILLED : Side x buffer not ready STATUS_CAL_DONE : All 6 buffers are full and ACC calibration done.	

NAME	**nvtGetAccOffset**	
SYNOPSIS	**void nvtGetAccOffset(float*)**	
Description	Get the calibration factor of ACC offset	
Parameters	**float* {3}**	[0]: X offset. [1]: Y offset. [2]: Z offset.
Return Value	void	

NAME	**nvtGetAccScale**

SYNOPSIS	**void nvtGetAccScale (float*)**	
Description	Get the calibration factor of ACC scale	
Parameters	**float* {3}**	[0]: X scale. [1]: Y scale. [2]: Z scale.
Return Value	void	

NAME	**nvtCalGyroInit**	
SYNOPSIS	**void nvtCalGyroInit(char axis);**	
Description	Application should call this function before processing GYRO scale calibration. Parameter 'axis' is the axis to be calibrated.	
Parameters	**char axis**	0: X 1: Y 2: Z
Return Value	void	

NAME	**nvtGyroScaleCalibrate**	
SYNOPSIS	**signed char nvtGyroScaleCalibrate(int8_t axis);**	
Description	Do the GYRO scale calibration by axis X,Y and Z. Parameter 'axis' can be 0 ~ 2.	
Parameters	**int8_t axis**	0: X 1: Y

		2: Z
Return Value	STATUS_GYRO_CAL_RUNNING: Scale calibration to axis running STATUS_GYRO_AXIS_CAL_DONE : Scale calibration to axis done	

NAME	**nvtGyroCenterCalibrate**	
SYNOPSIS	**signed char nvtGyroCenterCalibrate(void)**	
Description	Do the GYRO center calibration. Just put sensor steady. Wait and check return value "STATUS_GYRO_CAL_DONE".	
Parameters	**int8_t axis**	0: X 1: Y 2: Z
Return Value	STATUS_GYRO_CAL_BEGINE: Center calibration begin STATUS_GYRO_CAL_RUNNING: Center calibration running STATUS_GYRO_CAL_DONE: Center calibration done	

NAME	**nvtGetCalibratedGYRO**	
SYNOPSIS	**void nvtGetCalibratedGYRO(float*);**	
Description	Get calibrated gyro from AHRS.	
Parameters	**float* {3}**	[0]: GYRO X (degree/second) [1]: GYRO Y (degree/second) [2]: GYRO Z (degree/second)
Return Value	void	

NAME	**nvtCalMAGBufferFill**	
SYNOPSIS	**signed char nvtCalMAGBufferFill(void);**	
Description	Direct the sampled MAG raw data into MAG calibration buffer in AHRS. When there are 120 samples in MAG calibration buffer, the sample is done and AHRS starts to calculate the calibration matrix for MAG.	
Parameters	**None**	
Return Value	STATUS_BUFFER_NOT_FILLED : Calibration buffer not filled yet. STATUS_CAL_DONE : Calibration buffer filled and calibration done.	

NAME	**nvtGetMagCalQFactor**	
SYNOPSIS	**uint8_t nvtGetMagCalQFactor(void)**	
Description	When MAG calibration is done. Call this function to get a calibration quality factor (0 ~ 255). A factor less than 5 is good and less than 10 is fine. You may consider to redo calibration when factor grater than 10.	
Parameters	**None**	
Return Value	0 ~ 255 0~5: Good 6~10: Fine 11~255: Bad	

NAME	**nvtGetAccZWithoutGravity**
SYNOPSIS	**void nvtGetAccZWithoutGravity(float *ZWithoutGravity, float *AccZMag);**

Description	Call this function to get the output Z without gravity effect and the magnitude of Z '.	
Parameters	**float *ZWithoutGravity, float *AccZMag**	ZWithoutGravity: (ACC_Z – Gravity_Z) AccZMag: (ACC_X* ACC_X + ACC_Y*ACC_Y+ACC_Z*ACC_Z)
Return Value	void	

NAME	**nvtGetMove**	
SYNOPSIS	**void nvtGetMove(float* Move)**	
Description	Call this function to get the move distance (cm) from AHRS ACC calculation.	
Parameters	**float * Move{3}**	Move[0]: X distance Move[1]: Y distance Move[2]: Z distance
Return Value	void	