Freescale Semiconductor
APPLICATION NOTE

Document Number: AN5017
Rev. 1, 9/2015

Aerospace, Android and Windows 8 Coordinate Systems

Mark Pedley and Michael Stanley

Contents

1 Introduction

1.1 Summary

This application note documents the three coordinate systems supported in the *Freescale Sensor Fusion Library*:

- Aerospace (an x=North, y=East, z=Down or NED standard)
- Android (an x=East, y=North, z=Up or ENU standard)
- Windows 8 (also an x=East, y=North, z=Up or ENU standard)

When developing for Android or Windows 8 systems, the user should use the required coordinate system. Otherwise, the Aerospace coordinate system is probably the best choice because its sequence of rotations and the ranges of its Euler angles result in a yaw/compass heading estimate that is more user friendly than that computed using the other two coordinate systems.

Specifically, a 180° roll in the Aerospace coordinate system does not change the yaw/compass heading while a 180° pitch rotation does change the yaw/compass heading by 180°. This matches the behavior of an aircraft where a 180° roll simply means that the aircraft is on the same compass heading but inverted. The opposite behavior is mandated for the Android and Windows 8 coordinate systems where a 180° roll introduces a 180° change in compass heading and a 180° pitch rotation has no effect on compass heading.

1.2 Software Functions

Sensor Fusion software functions

Function	Description	Section

`void fNEDAnglesDegFromRotationMatrix (float R[][3], float *pfPhiDeg, float *pfTheDeg, float *pfPsiDeg, float *pfRhoDeg, float *pfChiDeg);`
 	
`void fAndroidAnglesDegFromRotationMatrix (float R[][3], float *pfPhiDeg, float *pfTheDeg, float *pfPsiDeg, float *pfRhoDeg, float *pfChiDeg);`
 	
`void fWin8AnglesDegFromRotationMatrix (float R[][3], float *pfPhiDeg, float *pfTheDeg, float *pfPsiDeg, float *pfRhoDeg, float *pfChiDeg);`
	Compute the Euler angles (roll, pitch, yaw) plus compass heading and tilt angles from Aerospace/NED, Android and Windows 8 orientation matrices.	2 (NED)
		3 (Android)
		4 (Windows 8)

1.3 Terminology

Term/Symbol	Definition
q	General quaternion $q = (q_0, q_1, q_2, q_3)$
$\boldsymbol{R}$	Rotation matrix
$\boldsymbol{R}_{NED}$	Aerospace (NED) rotation matrix
$\boldsymbol{R}_{Android}$	Android rotation matrix
$\boldsymbol{R}_{Win8}$	Windows 8 rotation matrix
$\boldsymbol{R}_x$	Rotation matrix about x-axis
$\boldsymbol{R}_y$	Rotation matrix about y-axis
$\boldsymbol{R}_z$	Rotation matrix about z-axis
θ	Pitch angle
ρ	Compass heading angle
ϕ	Roll angle
χ	Tilt angle
ψ	Yaw angle

1.4 Equivalency of Gravity and Acceleration

Accelerometers measure linear acceleration minus the gravity component in each axis. This is a consequence of basic Physics which states the equivalence between the forces experienced when at rest in a gravitational field and when accelerating in the opposite direction to gravity.

For example, an observer sitting on a chair at rest on earth with a 1*g* downward-pointing gravitational field experiences exactly the same force as when sitting in a spaceship remote from any gravitational field accelerating at 1*g* in the opposite direction.

1.5 Differences in Coordinate Systems

The Aerospace coordinate system defines the direction of axes in the zero rotation orientation as being x=North, y=East and z=Down (or NED). The Android and Windows 8 coordinate systems define the direction of axes in the same zero rotation orientation to be x=East, y=North, z=Up (or ENU).

An additional complication is that, whereas all accelerometers are natively acceleration positive in their outputs, the axes in the Aerospace and Windows 8 coordinate systems are specified to be gravity positive meaning that an axis measures +1*g* when pointed downward and aligned with gravity. The Android coordinate system is acceleration positive meaning that an axis measures +1*g* when pointed upwards and aligned with the equivalent 1*g* acceleration in the opposite direction to gravity.

The Aerospace and Windows 8 coordinate systems define rotations in the usual mathematical sense where a rotation is positive when it has a clockwise perspective when viewed in the increasing direction of an axis. Rotations in the Android coordinate system have the opposite sign.

0 summarizes these differences.

Coordinate system comparison

Item	Aerospace (NED)	Android (ENU)	Windows 8 (ENU)
Axes alignment	XYZ = NED	XYZ = ENU	XY = ENU
Accelerometer sign	Gravity-Acceleration +1*g* when axis is down –1*g* when axis is up	Acceleration-Gravity +1*g* when axis is up –1*g* when axis is down	Gravity-Acceleration +1*g* when axis is down –1*g* when axis is up
Accelerometer reading when flat	G[Z] = +1*g*	G[Z] = +1*g*	G[Z] = –1*g*
Direction of positive rotation	Clockwise	Anticlockwise	Clockwise
Compass heading ρ and yaw angle ψ	$\rho = \psi$	$\rho = \psi$	$\rho = 360° - \psi$

Refer to Sections 2.1, 3.1 and 4.1 for definitions of the axes, the sign standard for the accelerometer and the definition of a positive rotation about any axis for the Aerospace, Android and Windows 8 coordinate systems.

1.6 Euler Angle Orientation Matrix

Euler angles (roll, pitch and yaw angles) have very poor mathematical properties in comparison with rotation matrices and quaternions and should be avoided. Euler angles are only used in Freescale's Sensor Fusion Library software as an output for human use since they are intuitively understandable.

One of the Euler angles' mathematical problems is that rotation matrices do not commute meaning that the order in which individual Euler angle rotations are applied is important. The same three Euler angles will give different rotation matrices when applied in different sequence. Simply specifying the roll, pitch and yaw Euler angles without specifying their sequence is meaningless.

Another problem is that Euler angles have discontinuities at certain orientations whereas a rotation matrix or rotation quaternion is always continuous and varies smoothly with rotation. Finally, at orientations termed 'gimbal lock' the Euler angles become ambiguous and oscillate even though the rotation matrix and quaternion are stable.

Since the three coordinate standards do require that Euler angles be computed, table 2 below lists the decomposition of the orientation matrix into three Euler angle rotation.

Sequence of Euler angle rotations

Item	Aerospace (NED)	Android (ENU)	Windows 8 (ENU)
Sequence of Euler angle rotations	$\boldsymbol{R} = \boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi)$	$\boldsymbol{R} = \boldsymbol{R}_x(\theta)\boldsymbol{R}_y(\phi)\boldsymbol{R}_z(\psi)$	$\boldsymbol{R} = \boldsymbol{R}_y(\phi)\boldsymbol{R}_x(\theta)\boldsymbol{R}_z(\psi)$

1.7 Gimbal Lock

Gimbal lock occurs in all three coordinate systems when the second rotation (pitch in Aerospace and Windows 8 systems and roll in the Android system) aligns the axes of the first and third Euler angle rotations. The number of degrees of freedom needed to define the orientation reduces from three Euler

angles to two. Any change in the first Euler rotation angle can be offset by a cancelling change in the third Euler rotation angle with the result that the two values oscillate unstably together.

Gimbal lock in strapdown navigation systems is, therefore, a mathematical instability that results from the decomposition of the orientation matrix or quaternion into three individual Euler angle rotations and is easily avoided by using orientation matrix or quaternion representations which are completely stable at gimbal lock orientations.

The Freescale Sensor Fusion Library uses orientation matrix and quaternion algebra throughout and computes Euler angles for human use as an output from the algorithms. The computed orientation is therefore just as stable at gimbal lock orientations as at other orientations. Any software fundamentally based on Euler angle rotations in preference to rotation matrix or quaternion will suffer from gimbal lock instability and mathematical discontinuities.

Gimbal lock defined by coordinate system

Item	**Aerospace (NED)**	**Android (ENU)**	**Windows 8 (ENU)**
Gimbal Lock	±90° pitch (y-axis)	±90° roll (y-axis)	±90° pitch (x-axis)

1.8 Euler Angle Discontinuities

Another problem with Euler angles representation of orientation is that, in addition to there being an infinite number of Euler angle solutions at gimbal lock orientations, there are, in general, two Euler angle solutions for every orientation. This forces one of the Euler angles to be constrained with a reduced range of –90° to +90° to eliminate one of the two solutions.

The presence of two solutions is easily demonstrated with the starting position of the device laid flat on the table and then applying these two rotation sequences:

- rotation of 180° in pitch
- rotation of 180° in yaw followed by a rotation of 180° in pitch

Both rotation sequences result in the same physical orientation and the same orientation matrix and quaternion.

The Aerospace coordinate system restricts the range of the pitch angle from –90° to +90° and the Android and Windows 8 coordinate systems restrict the range of roll angle from –90° to +90°. Although this restriction results in just one Euler angle solution at all orientations except at gimbal lock, it creates the unfortunate mathematical side effect of introducing discontinuities in the Euler angles at –90° and +90° roll angles in the Android and Windows 8 coordinate systems.

Although this side effect occurs at the same orientation as gimbal lock in the Aerospace and Android coordinate systems, in the Windows 8 coordinate system it is a different phenomenon. This is shown when the side effect occurs at an orientation (–90° and +90° roll) that is different from gimbal lock (–90° and +90° pitch) in the Windows 8 coordinate system.

Euler angles are mathematically troublesome and must be avoided in the inner workings of strapdown Sensor Fusion Library software. The Euler angles' only role is providing output to humans as specified by the Android, Windows 8 or other standard.

0 summarizes the Euler angle discontinuities in each coordinate system. Sections 2 through 4 present more information on the meanings of the various table entries.

Euler angle discontinuities in coordinate systems

Item	Aerospace (NED)	Android (ENU)	Windows 8 (ENU)
Ranges of Euler angles	0° ≤ Yaw $\psi(z)$ < 360° –90° ≤ Pitch $\theta(y)$ < 90° –180° ≤ Roll $\phi(x)$ < 180°	0° ≤ Yaw $\psi(z)$ < 360° –90° ≤ Roll $\phi(y)$ < 90° –180° ≤ Pitch $\theta(x)$ < 180°	0° ≤ Yaw $\psi(z)$ < 360° –180° ≤ Pitch $\theta(x)$ < 180° –90° ≤ Roll $\phi(y)$ < 90°
Equivalent matrix	$\boldsymbol{R}_x(\phi+\pi)\boldsymbol{R}_y(\pi-\theta)\boldsymbol{R}_z(\psi+\pi)$	$\boldsymbol{R}_x(\theta+\pi)\boldsymbol{R}_y(\pi-\phi)\boldsymbol{R}_z(\psi+\pi)$	$\boldsymbol{R}_y(\phi-\pi)\boldsymbol{R}_x(\pi-\theta)\boldsymbol{R}_z(\psi+\pi)$
Behavior during ±180 degree roll rotation	Roll is continuous in range –180° to 180°. No change in yaw or compass angle.	Roll is continuous, increasing to 90° and then decreasing or decreasing to –90° and then increasing. Pitch and yaw have 180° discontinuities at ±90° roll.	180° jump in roll, pitch, yaw and compass as the roll angle passes 90° and –90°.
Behavior during ±180 degree pitch rotation	Pitch is continuous, increasing to 90° and then decreasing or decreasing to –90° and then increasing. Roll and yaw have 180° discontinuity at ±90° pitch.	Smooth changes in pitch. No change in roll, yaw or compass.	Smooth changes in pitch. No change in roll, yaw or compass.

2 Aerospace (NED) Coordinate System

2.1 Axes Definitions

The Aerospace coordinate system is shown in Figure 1. It is an x=North, y=East, z=Down (termed NED) coordinate system, meaning that when the product is positioned in its default orientation, flat on the table and pointed northward, the x-axis points north, the y-axis points east and the z-axis points down.

The sign of rotations about the *x*, *y* and *z* axes is defined in the normal conventions as a clockwise rotation and is deemed positive when looking along the increasing direction of the axis. Roll, pitch and yaw rotations have their normal meaning, and therefore, correspond to rotations about the x, y and z axes, respectively. The yaw angle ψ equals the compass angle ρ.

The Aerospace coordinate system is gravity (not acceleration) positive, meaning that the output of any accelerometer channel is positive when pointing downward and aligned with gravity. The accelerometer z-axis reading is therefore +1*g* when the device is flat and upright.

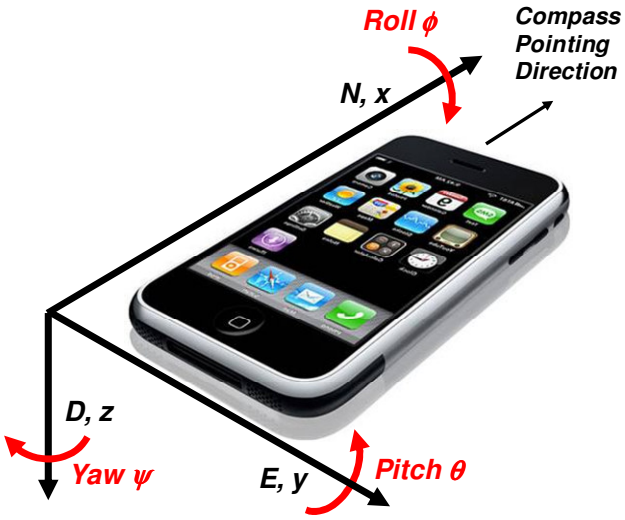

Figure 1. Aerospace (NED) coordinate system

2.2 Orientation Matrix in Terms of Euler Angles

The sequence of Euler angle rotations in the Aerospace coordinate system is yaw followed by pitch and finally roll. The roll rotation ϕ is about the x-axis, the pitch rotation θ about the y-axis and the yaw rotation ψ about the z-axis. The individual rotation matrices are:

$$\boldsymbol{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{pmatrix} \quad (1)$$

$$\boldsymbol{R}_y(\theta) = \begin{pmatrix} cos\,\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{pmatrix} \quad (2)$$

$$\boldsymbol{R}_z(\psi) = \begin{pmatrix} cos\,\psi & sin\psi & 0 \\ -sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3)$$

The composite Aerospace rotation matrix using the specified rotation sequence is:

$$\boldsymbol{R}_{NED} = \boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{pmatrix}\begin{pmatrix} cos\,\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{pmatrix}\begin{pmatrix} cos\,\psi & sin\psi & 0 \\ -sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (4)$$

$$\Rightarrow \boldsymbol{R}_{NED} = \begin{pmatrix} cos\,\theta\,cos\,\psi & cos\,\theta\,sin\psi & -sin\theta \\ cos\,\psi\,sin\theta sin\phi - cos\,\phi\,sin\psi & cos\,\phi\,cos\,\psi + sin\theta sin\phi sin\psi & cos\,\theta\,sin\phi \\ cos\phi\,cos\,\psi\,sin\theta + sin\phi sin\psi & cos\,\phi\,sin\theta sin\psi - cos\,\psi\,sin\phi & cos\,\theta\,cos\phi \end{pmatrix} \quad (5)$$

2.3 Orientation Quaternion in Terms of Euler Angles

The individual Euler angle rotation quaternions in the Aerospace coordinate system are:

$$q_x(\phi) = cos\left(\frac{\phi}{2}\right) + \boldsymbol{i}sin\left(\frac{\phi}{2}\right) \quad \textbf{(6)}$$

$$q_y(\theta) = cos\left(\frac{\theta}{2}\right) + \boldsymbol{j}sin\left(\frac{\theta}{2}\right) \quad \textbf{(7)}$$

$$q_z(\psi) = cos\left(\frac{\psi}{2}\right) + \boldsymbol{k}sin\left(\frac{\psi}{2}\right) \quad \textbf{(8)}$$

The Aerospace rotation quaternion $q_{zyx} = q_z(\psi)q_y(\theta)q_x(\phi)$ evaluates to:

$$q_{zyx} = q_z(\psi)q_y(\theta)q_x(\phi) = \left\{cos\left(\frac{\psi}{2}\right) + \boldsymbol{k}sin\left(\frac{\psi}{2}\right)\right\}\left\{cos\left(\frac{\theta}{2}\right) + \boldsymbol{j}sin\left(\frac{\theta}{2}\right)\right\}\left\{cos\left(\frac{\phi}{2}\right) + \boldsymbol{i}sin\left(\frac{\phi}{2}\right)\right\} \quad \textbf{(9)}$$

$$\begin{aligned} &= \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\} \\ &\quad + \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right)\right\}\boldsymbol{i} \\ &\quad + \left\{cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{j} \\ &\quad + \left\{sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{k} \end{aligned} \quad \textbf{(10)}$$

The elements of the Aerospace quaternion q_{zyx} are, therefore:

$$q_0 = cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) \quad \textbf{(11)}$$

$$q_1 = cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) \quad \textbf{(12)}$$

$$q_2 = cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) \quad \textbf{(13)}$$

$$q_3 = sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) \quad \textbf{(14)}$$

2.4 Gimbal Lock

Gimbal lock occurs in the Aerospace coordinate system when the second pitch rotation aligns the first yaw and third roll rotations. This occurs at 90° pitch upward and 90° pitch downward. The number of

degrees of freedom in the Euler angle description of the orientation reduces from three to two and the roll and yaw angles oscillate with only their sum and difference defined.

At gimbal lock $\theta = 90deg$ and $\theta = -90deg$, equation (5) simplifies to:

$$\boldsymbol{R}_{NED} = \begin{pmatrix} 0 & 0 & -1 \\ -sin(\psi - \phi) & cos(\psi - \phi) & 0 \\ cos(\psi - \phi) & sin(\psi - \phi) & 0 \end{pmatrix} for\ \theta = 90deg \quad \textbf{(15)}$$

$$\boldsymbol{R}_{NED} = \begin{pmatrix} 0 & 0 & 1 \\ -sin(\psi + \phi) & cos(\psi + \phi) & 0 \\ -cos(\psi + \phi) & -sin(\psi + \phi) & 0 \end{pmatrix} for\ \theta = -90deg \quad \textbf{(16)}$$

The orientation matrix and quaternion are stable but only the differenced angle $\psi - \phi$ or the summed angle $\psi + \phi$ can be determined at the two gimbal lock orientations.

2.5 Euler Angle Discontinuities

The Aerospace rotation matrix $\boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi)$ is equal to the matrix $\boldsymbol{R}_x(\phi + \pi)\boldsymbol{R}_y(\pi - \theta)\boldsymbol{R}_z(\psi + \pi)$, where π radians correspond to 180°. By direct evaluation:

$$\boldsymbol{R}_x(\phi + \pi)\boldsymbol{R}_y(\pi - \theta)\boldsymbol{R}_z(\psi + \pi)$$

$$= \begin{pmatrix} -cos(\pi - \theta)\, cos\,\psi & -cos(\pi - \theta)\, sin\psi & -sin(\pi - \theta) \\ cos\,\psi\, sin(\pi - \theta) sin\phi - cos\,\phi\, sin\psi & cos\,\phi\, cos\,\psi + sin(\pi - \theta) sin\phi sin\psi & -cos(\pi - \theta)\, sin\phi \\ cos\phi\, cos\,\psi\, sin(\pi - \theta) + sin\phi sin\psi & cos\,\phi\, sin(\pi - \theta) sin\psi - cos\,\psi\, sin\phi & -cos(\pi - \theta)\, cos\phi \end{pmatrix} \quad \textbf{(17)}$$

$$= \begin{pmatrix} cos\,\theta\, cos\,\psi & cos\,\theta\, sin\psi & -sin\theta \\ cos\,\psi\, sin\theta sin\phi - cos\,\phi\, sin\psi & cos\,\phi\, cos\,\psi + sin\theta sin\phi sin\psi & cos\,\theta\, sin\phi \\ cos\phi\, cos\,\psi\, sin\theta + sin\phi sin\psi & cos\,\phi\, sin\theta sin\psi - cos\,\psi\, sin\phi & cos\,\theta\, cos\phi \end{pmatrix} = \boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi) \quad \textbf{(18)}$$

This is the mathematical identity equivalent to the statement made in Section 1.8 that there are two Euler angle solutions to each orientation. This statement can be reworded as: An alternative to a rotation of ψ in yaw then θ in pitch and finally by ϕ in roll, one can arrive at the same orientation by rotating 180° plus ψ in yaw, then 180° minus θ in pitch and finally by 180° plus ϕ in roll.

The Aerospace coordinate system removes one of the two solutions by restricting the pitch angle θ to the range –90° to +90°. Equations (17) and (18) state that whenever the pitch angle is above 90° or less than –90° the roll angle should increase by 180° (the $\boldsymbol{R}_x(\phi + \pi)$ term), the pitch angle should be negated, 180° added (the $\boldsymbol{R}_y(\pi - \theta)$ term) and the yaw/compass heading increased by 180° (the $\boldsymbol{R}_z(\psi + \pi)$ term). These discontinuities are shown in Figure 2 through Figure 4

Figure 2 shows the Euler angles as the device is rotated through 360° in roll from the default starting position of being flat and pointed northward. Neither the pitch nor yaw angles change and the roll discontinuity from –180° to +180° is a consequence of modulo 360° arithmetic and is quite acceptable. The average user of an e-compass will agree that if a smartphone is pointed north and then given a roll rotation of 180°, then it is still pointed north.

Aerospace (NED) Coordinate System

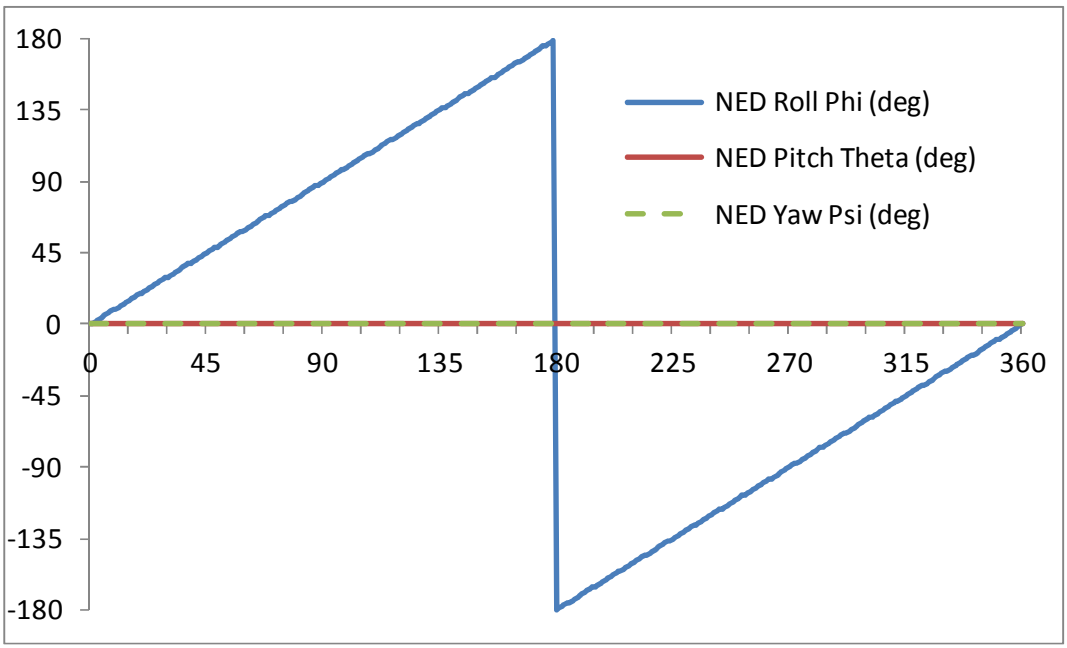

Figure 2. 360° roll rotation at zero pitch and yaw in the NED coordinate system

Figure 3 shows the Euler angles as the PCB is rotated 360° in pitch from the same starting position. The pitch angle reaches 90° and then starts smoothly decreasing without a discontinuity until it reaches –90° and then starts increasing smoothly. The yaw/compass angle flips by 180° at the 90° and –90° pitch angles, but this is reasonable ergonomic behavior. For the same reasons discussed above, the average smartphone user will consider a 180° pitch rotation to flip the yaw/compass heading 180°.

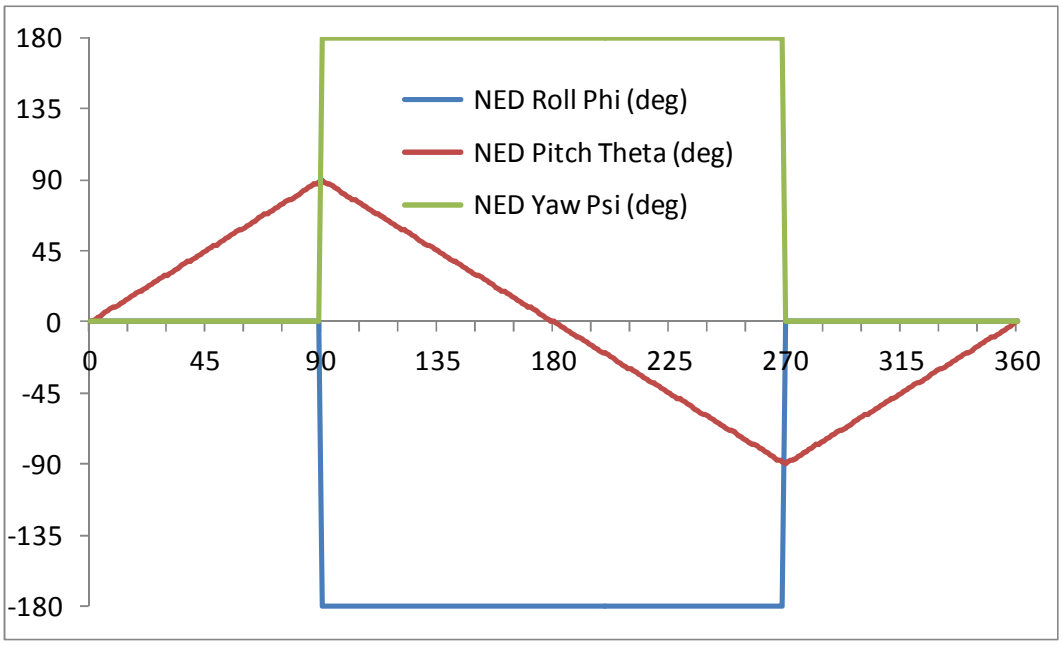

Figure 3. 360° pitch rotation at zero roll and yaw in the NED coordinate system

Figure 4 shows the yaw/compass angle behavior as the PCB is rotated 360° in yaw while remaining flat. The yaw/compass heading increases smoothly with the acceptable modulo 360° discontinuity when pointed northward.

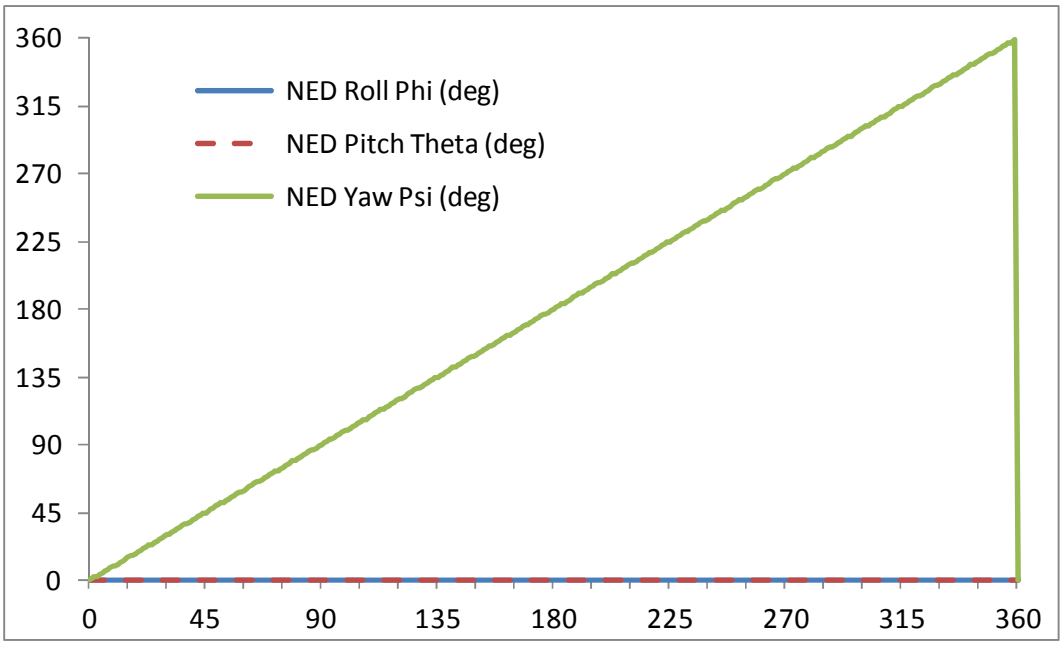

Figure 4. 360° yaw rotation at zero roll and pitch in the NED coordinate system

The behavior illustrated in Figure 4 is the justification for the comment made in Section 1 that the Aerospace coordinate system is the most ergonomically reasonable and should be used in preference unless specifically developing for the Android and Windows 8 standards.

2.6 Calculation of Euler Angles from Rotation Matrix

This section documents function `fNEDAnglesDegFromRotationMatrix` which computes the Euler angles from the Aerospace rotation matrix as defined in equation (5).

The solution for the three Aerospace Euler angles is:

$$\phi = tan^{-1}\left(\frac{R_{yz}}{R_{zz}}\right), -180 \leq \phi < 180\ deg \quad \textbf{(19)}$$

$$\theta = sin^{-1}(-R_{xz}), -90 \leq \theta \leq 90\ deg \quad \textbf{(20)}$$

$$\psi = tan^{-1}\left(\frac{R_{xy}}{R_{xx}}\right), 0 \leq \psi < 360\ deg \quad \textbf{(21)}$$

The Aerospace compass heading angle ρ always equals the yaw angle ψ:

$$\rho = \psi \quad \textbf{(22)}$$

At gimbal lock, equations (5) and (6) give:

$$tan(\psi - \phi) = \left(\frac{R_{zy}}{R_{yy}}\right) for\ \theta = 90deg \quad \textbf{(23)}$$

$$tan(\psi + \phi) = \left(\frac{-R_{zy}}{R_{yy}}\right) for\ \theta = -90deg \quad \textbf{(24)}$$

The tilt angle from vertical χ can be determined from the scalar product of the rotated gravity vector and the downwards z-axis, giving:

$$cos\chi = \left\{\boldsymbol{R}_{NED}\begin{pmatrix}0\\0\\1\end{pmatrix}\right\}\cdot\begin{pmatrix}0\\0\\1\end{pmatrix} = \begin{pmatrix}R_{xz}\\R_{yz}\\R_{zz}\end{pmatrix}\cdot\begin{pmatrix}0\\0\\1\end{pmatrix} = R_{zz} = cos\,\theta\,cos\phi \quad \textbf{(25)}$$

3 Android Coordinate System

3.1 Axes Definitions

The reference for this section is the Android specification available at:

http://developer.android.com/reference/android/hardware/SensorEvent.html

Android: *"The coordinate space is defined relative to the screen of the phone in its default orientation. The axes are not swapped when the device's screen orientation changes. The OpenGL ES coordinate system is used. The origin is in the lower-left corner with respect to the screen, with the x-axis horizontal and pointing right, the y-axis vertical and pointing up and the z-axis pointing outside the front face of the screen. In this system, coordinates behind the screen have negative z values."*

The Android sensor coordinate system is shown in Figure 5. It is an x=East, y=North, z=Up (termed ENU) coordinate system meaning that when the product is in its default orientation (lying flat on the table and pointed northward) the x-axis points east, the y-axis points north and the z-axis points up.

The signs of rotations about the x, y and z axes are defined as opposite to normal mathematical sense. A counterclockwise rotation is deemed positive when looking along the increasing direction of the axis. Roll, pitch and yaw rotations have their normal meaning and, therefore, correspond to rotations about the y, x and z axes respectively. The yaw angle ψ equals the compass angle ρ.

The Android coordinate system is acceleration (not gravity) positive, meaning that the output of any accelerometer channel is positive when pointing upward and aligned against gravity. The accelerometer z-axis reading is therefore +1*g* when the product is flat and upright. This is the same as the Aerospace coordinate system as a result of two cancelling sign changes: i) the z axes point in opposite directions and ii) Aerospace is gravity positive and Android is acceleration positive.

Android: *"When the device lies flat on a table, the acceleration value is +9.81, which corresponds to the acceleration of the device (0 m/sec^2) minus the force of gravity."*

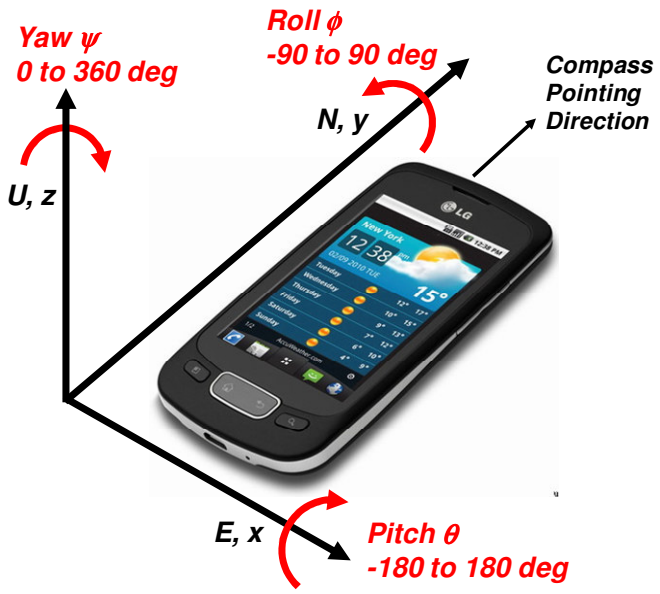

Figure 5. Android coordinate system

3.2 Orientation Matrix in Terms of Euler Angles

The sequence of Euler angle rotations in the Android coordinate system does not appear to be documented in the standard. Investigation of the behavior of Android tablets and phones indicates that most use i) first yaw ii) secondly roll and iii) finally pitch. The roll rotation ϕ is about the y axis, the pitch rotation θ about the x axis and the yaw rotation ψ about the z axis. The individual rotation matrices are:

$$\boldsymbol{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix} \tag{26}$$

$$\boldsymbol{R}_y(\phi) = \begin{pmatrix} cos\,\phi & 0 & sin\phi \\ 0 & 1 & 0 \\ -sin\phi & 0 & cos\phi \end{pmatrix} \tag{27}$$

$$\boldsymbol{R}_z(\psi) = \begin{pmatrix} cos\,\psi & -sin\psi & 0 \\ sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{28}$$

The composite Android rotation matrix using the specified rotation sequence is:

$$\boldsymbol{R}_{Android} = \boldsymbol{R}_x(\theta)\boldsymbol{R}_y(\phi)\boldsymbol{R}_z(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix}\begin{pmatrix} cos\,\phi & 0 & sin\phi \\ 0 & 1 & 0 \\ -sin\phi & 0 & cos\phi \end{pmatrix}\begin{pmatrix} cos\,\psi & -sin\psi & 0 \\ sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{29}$$

$$= \begin{pmatrix} cos\,\phi\,cos\,\psi & -cos\phi sin\psi & sin\phi \\ cos\,\theta\,sin\psi + cos\,\psi\,sin\phi sin\theta & cos\,\psi\,cos\,\theta - sin\phi sin\psi sin\theta & -cos\,\phi\,sin\theta \\ -cos\,\psi\,cos\,\theta\,sin\phi + sin\psi sin\theta & cos\,\theta\,sin\phi sin\psi + cos\,\psi\,sin\theta & cos\,\phi\,cos\,\theta \end{pmatrix} \tag{30}$$

3.3 Orientation Quaternion in Terms of Euler Angles

The individual Euler angle rotation quaternions in the Android coordinate system are:

$$q_x(\theta) = cos\left(\frac{\theta}{2}\right) - \boldsymbol{i}sin\left(\frac{\theta}{2}\right) \tag{31}$$

$$q_y(\phi) = cos\left(\frac{\phi}{2}\right) - \boldsymbol{j}sin\left(\frac{\phi}{2}\right) \tag{32}$$

$$q_z(\psi) = cos\left(\frac{\psi}{2}\right) - \boldsymbol{k}sin\left(\frac{\psi}{2}\right) \tag{33}$$

The Android rotation quaternion $q_{zyx} = q_z(\psi)q_y(\phi)q_x(\theta)$ evaluates to:

$$q_{zyx} = q_z(\psi)q_y(\phi)q_x(\theta) = \left\{cos\left(\frac{\psi}{2}\right) - \boldsymbol{k}sin\left(\frac{\psi}{2}\right)\right\}\left\{cos\left(\frac{\phi}{2}\right) - \boldsymbol{j}sin\left(\frac{\phi}{2}\right)\right\}\left\{cos\left(\frac{\theta}{2}\right) - \boldsymbol{i}sin\left(\frac{\theta}{2}\right)\right\} \tag{34}$$

$$\begin{aligned} &= \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\} \\ &\quad - \left\{cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{i} \\ &\quad - \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right)\right\}\boldsymbol{j} \\ &\quad - \left\{sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{k} \end{aligned} \tag{35}$$

The elements of the Android quaternion q_{zyx} are therefore:

$$q_0 = cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) \tag{36}$$

$$q_1 = -cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) \tag{37}$$

$$q_2 = -cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) \tag{38}$$

$$q_3 = -sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) \tag{39}$$

3.4 Gimbal Lock

Gimbal lock occurs in the Android coordinate system when the second roll rotation aligns the first yaw and third pitch rotations. This occurs at +90° roll and –90° roll. The number of degrees of freedom in the

Euler angle description of the orientation, therefore, reduces from three to two and the pitch and yaw angles oscillate with only their sum and difference defined.

At gimbal lock $\phi = 90deg$, equation (30) simplifies to:

$$\boldsymbol{R}_{Android} = \begin{pmatrix} 0 & 0 & 1 \\ sin(\psi+\theta) & cos(\psi+\theta) & 0 \\ -cos(\psi+\theta) & sin(\psi+\theta) & 0 \end{pmatrix} for\ \phi = 90deg \quad \textbf{(40)}$$

At gimbal lock $\phi = -90deg$, equation (30) simplifies to:

$$\boldsymbol{R}_{Android} = \begin{pmatrix} 0 & 0 & -1 \\ sin(\psi-\theta) & cos(\psi-\theta) & 0 \\ cos(\psi-\theta) & -sin(\psi-\theta) & 0 \end{pmatrix} for\ \phi = -90deg \quad \textbf{(41)}$$

The orientation matrix and quaternion are stable, but the summed angle $\psi + \theta$ or the differences angle $\psi - \theta$ can be determined at the two gimbal lock orientations.

3.5 Euler Angle Discontinuities

The Android rotation matrix $\boldsymbol{R}_x(\theta)\boldsymbol{R}_y(\phi)\boldsymbol{R}_z(\psi)$ is equivalent to $\boldsymbol{R}_x(\theta+\pi)\boldsymbol{R}_y(\pi-\phi)\boldsymbol{R}_z(\psi+\pi)$. By direct evaluation:

$\boldsymbol{R}_x(\theta+\pi)\boldsymbol{R}_y(\pi-\phi)\boldsymbol{R}_z(\psi+\pi)$

$$= \begin{pmatrix} -cos(\pi-\phi)\,cos\,\psi & cos(\pi-\phi)sin\psi & sin(\pi-\phi) \\ cos\,\theta\,sin\psi + cos\,\psi\,sin(\pi-\phi)sin\theta & cos\,\psi\,cos\,\theta - sin(\pi-\phi)sin\psi sin\theta & cos(\pi-\phi)\,sin\theta \\ -cos\,\psi\,cos\,\theta\,sin(\pi-\phi) + sin\psi sin\theta & cos\,\theta\,sin(\pi-\phi)sin\psi + cos\,\psi\,sin\theta & -cos(\pi-\phi)\,cos\,\theta \end{pmatrix} \quad \textbf{(42)}$$

$$= \begin{pmatrix} cos\,\phi\,cos\,\psi & -cos\phi sin\psi & sin\phi \\ cos\,\theta\,sin\psi + cos\,\psi\,sin\phi sin\theta & cos\,\psi\,cos\,\theta - sin\phi sin\psi sin\theta & -cos\,\phi\,sin\theta \\ -cos\,\psi\,cos\,\theta\,sin\phi + sin\psi sin\theta & cos\,\theta\,sin\phi sin\psi + cos\,\psi\,sin\theta & cos\,\phi\,cos\,\theta \end{pmatrix} = \boldsymbol{R}_x(\theta)\boldsymbol{R}_y(\phi)\boldsymbol{R}_z(\psi) \quad \textbf{(43)}$$

Equations (42) and (43) show that there are two solutions for the Android Euler angles for a given rotation matrix. If ϕ, θ and ψ are one solution for the roll, pitch and yaw angles for a given orientation matrix, then so is the solution given by roll angle 180° minus ϕ, pitch angle 180° plus θ and yaw angle 180° plus ψ.

The Android specification removes the two solutions by restricting the roll angle θ to the range –90° to +90°. From the Android specification, where all values are angles in degrees.

Android: ***"values[0]:*** *Azimuth, angle between the magnetic north direction and the y-axis, around the z-axis (0 to 359). 0=North, 90=East, 180=South, 270=West*

values[1]: *Pitch, rotation around x-axis (–180° to 180°), with positive values when the z-axis moves toward the y-axis.*

values[2]: *Roll, rotation around y-axis (–90° to 90°), with positive values when the x-axis moves toward the z-axis."*

Equations (42) and (43) state that whenever the roll angle is above 90° or less than -90° then the pitch angle should increase by 180° (the $\boldsymbol{R}_x(\theta+\pi)$ term), the roll angle should be negated and 180° added

(the $\boldsymbol{R}_y(\pi - \phi)$ term) and the yaw/compass heading should increase by 180° (the $\boldsymbol{R}_z(\psi + \pi)$ term). These discontinuities are shown in Figure 6 through Figure 8.

Figure 6 shows the Euler angles as the PCB is rotated through 360° in roll from the default starting position of being flat and pointed northward. The roll angle reaches 90° and then starts smoothly decreasing without a discontinuity until it reaches –90° and then starts increasing smoothly. The yaw/compass angle flips by 180° at the 90° and –90° roll angles.

Figure 7 shows the Euler angles as the PCB is rotated 360° in pitch from the same starting position. The pitch angle increases smoothly with the acceptable modulo 360° discontinuity between the equivalent angles of –180° and +180°.

Figure 8 shows the yaw/compass angle behavior as the PCB is rotated 360° in yaw while remaining flat. The yaw/compass heading increases smoothly with the acceptable modulo 360° discontinuity when pointed northward.

The 180° compass heading change when the Android roll angle passes through –90° or +90° is arguably not ideal from an ergonomic standpoint. It can be stated that, given an initial starting point pointing north, then:

- After a 180° roll rotation, it is still pointing northward (Android says it now points south)
- After a 180° pitch rotation, it is pointing southward (Android says it is still pointing north).

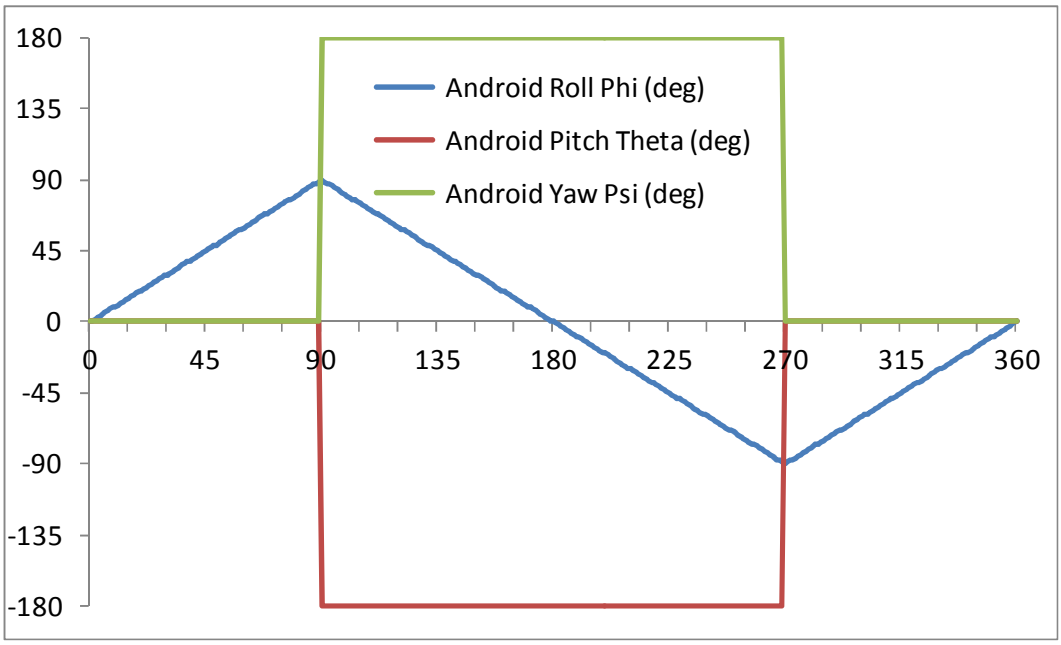

Figure 6. 360° roll rotation at zero pitch and yaw in the Android coordinate system

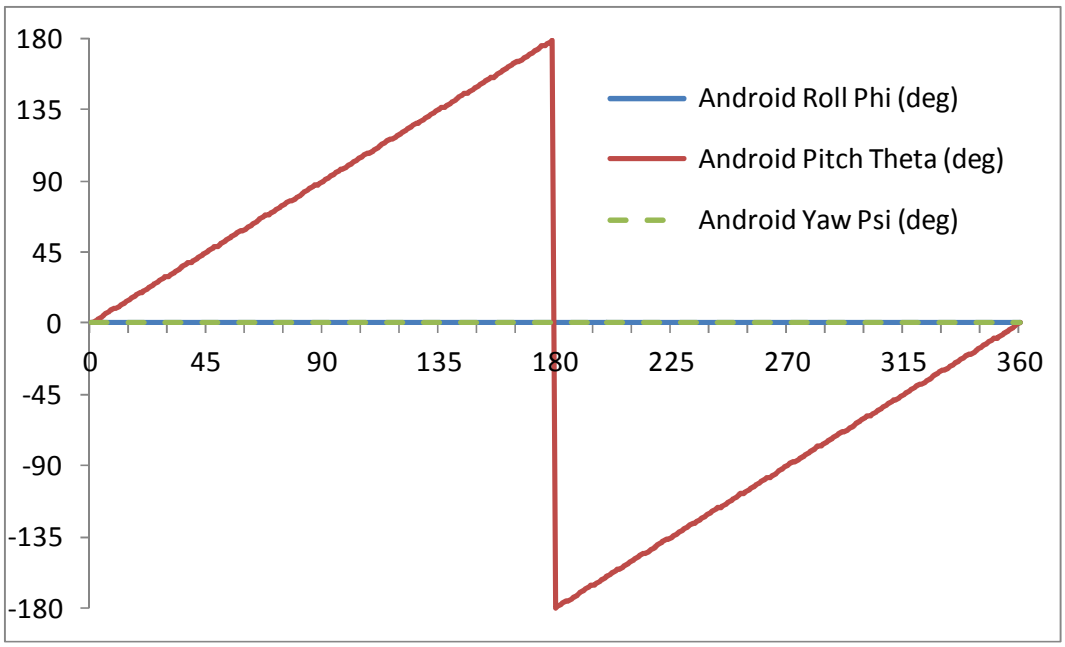

Figure 7. 360° pitch rotation at zero roll and yaw in the Android coordinate system

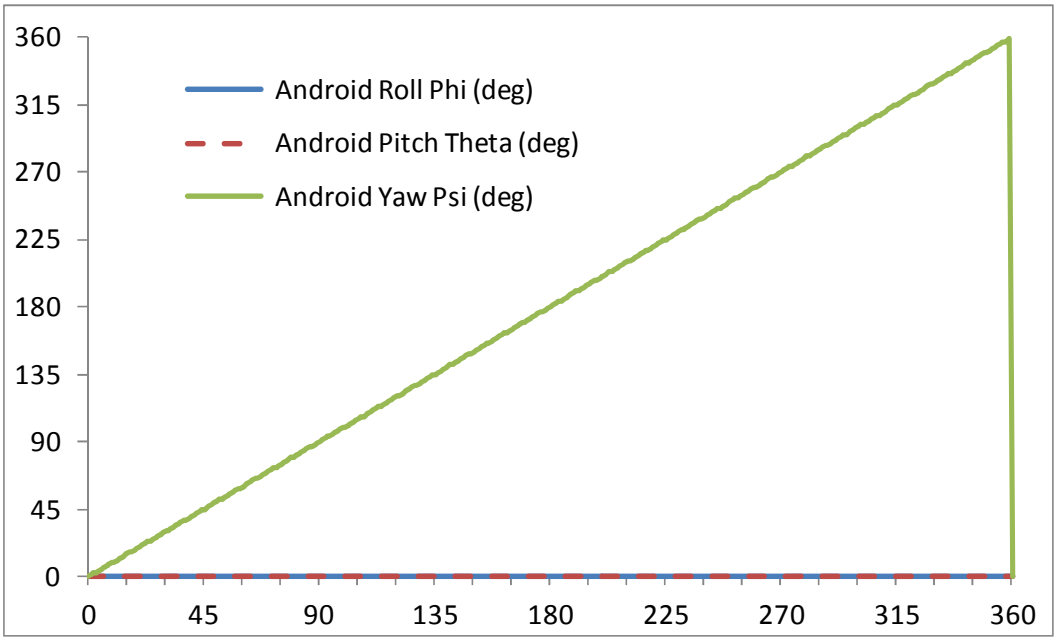

Figure 8. 360° yaw rotation at zero roll and pitch in the Android coordinate system

3.6 Calculation of Euler Angles from Rotation Matrix

This section documents function `fAndroidAnglesDegFromRotationMatrix`, which computes the Euler angles from the Android rotation matrix as defined in equation (30).

The solution for the three Android Euler angles is:

$$\phi = sin^{-1}(R_{xz}), -90 \leq \phi < 90\ deg \quad \textbf{(44)}$$

$$\theta = tan^{-1}\left(\frac{-R_{yz}}{R_{zz}}\right), -180 \leq \theta < 180\ deg \quad \textbf{(45)}$$

$$\psi = tan^{-1}\left(\frac{-R_{xy}}{R_{xx}}\right), 0 \leq \psi < 360\ deg \quad \textbf{(46)}$$

The Android compass heading angle ρ always equals the yaw angle ψ:

$$\rho = \psi \quad \textbf{(47)}$$

At gimbal lock, equations (40) and (41) give:

$$tan(\psi + \theta) = \left(\frac{R_{yx}}{R_{yy}}\right) for\ \phi = 90deg \quad \textbf{(48)}$$

$$tan(\psi - \theta) = \left(\frac{R_{yx}}{R_{yy}}\right) for\ \phi = -90deg \quad \textbf{(49)}$$

The tilt angle from vertical χ can be determined from the scalar product of the rotated gravity vector and the downwards z-axis giving:

$$cos\chi = \left\{\boldsymbol{R}_{Android}\begin{pmatrix}0\\0\\1\end{pmatrix}\right\}\cdot\begin{pmatrix}0\\0\\1\end{pmatrix} = \begin{pmatrix}R_{xz}\\R_{yz}\\R_{zz}\end{pmatrix}\cdot\begin{pmatrix}0\\0\\1\end{pmatrix} = R_{zz} = cos\,\theta\,cos\phi \quad \textbf{(50)}$$

4 Windows 8 Coordinate System

4.1 Axes Definitions

The reference for this section is the Microsoft specification: "*Integrating Motion and Orientation Sensors*" available for download at:

http://msdn.microsoft.com/en-us/library/windows/hardware/dn642102(v=vs.85).aspx

The Windows 8 coordinate system is shown in Figure 9. It is an x=East, y=North, z=Up (termed ENU) coordinate system, similar to the Android coordinate system.

The sign of rotations about the x, y and z axes is opposite to Android and, therefore, compatible with normal mathematical usage. A consequence is that the Windows 8 compass heading angle ρ has the opposite sense to the Windows 8 yaw angle ψ and ρ=360°–ψ.

The Windows 8 coordinate system is gravity (not acceleration) positive, meaning that the output of any accelerometer channel is negative when pointing up and aligned against gravity. The accelerometer z-axis reading is, therefore, –1*g* when the product is flat and upright.

Microsoft: *"a. Lay the device on a flat surface with the screen pointing upward, the acceleration values should be the following: X = ~0.0, Y = ~0.0, Z = ~-1.0*

b. Hold the device with the right hand side pointing upward, the acceleration values should be the following: X = ~-1.0, Y = ~0.0, Z = ~0.0

c. Hold the device with the top of the screen pointing upward, the acceleration values should be the following: X = ~0.0, Y = ~-1.0, Z = ~0.0"

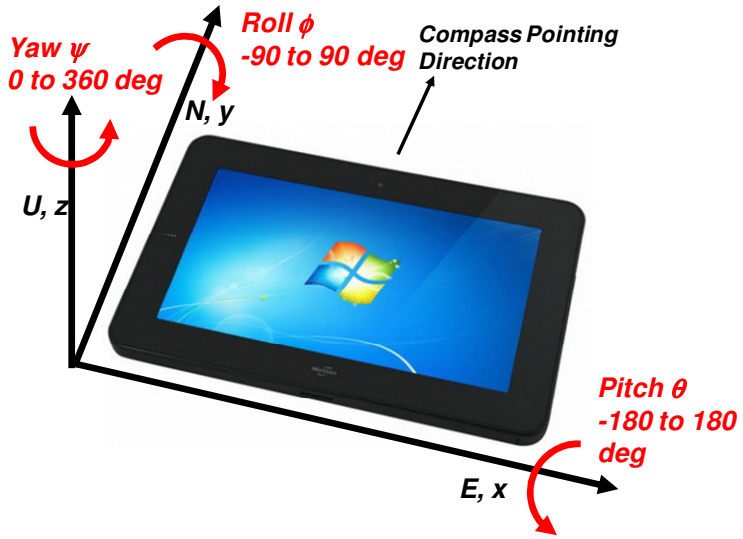

Figure 9. Windows 8 coordinate system

4.2 Orientation Matrix in Terms of Euler Angles

The order of the rotations in Windows 8 matches the Aerospace standard but not Android. The yaw rotation is first, followed by pitch rotation and then followed by a roll rotation.

Microsoft: *"In order to properly describe the orientation of the device in 3Dspace, the Euler angles are applied in the following order, starting with the device lying on a flat surface and the +y-axis pointing due north (toward the North Pole):*

1. *The device is rotated by the Yaw angle about its z-axis (opposite rotation direction about z-axis compared to heading).*
2. *The device is rotated by the Pitch angle about its x-axis.*
3. *The device is rotated by the Roll angle about its y-axis."*

The individual Windows rotation matrices are:

$$\boldsymbol{R_x}(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{pmatrix} \tag{51}$$

$$\boldsymbol{R_y}(\phi) = \begin{pmatrix} cos\,\phi & 0 & -sin\phi \\ 0 & 1 & 0 \\ sin\phi & 0 & cos\phi \end{pmatrix} \tag{52}$$

$$\boldsymbol{R_z}(\psi) = \begin{pmatrix} cos\,\psi & sin\psi & 0 \\ -sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{53}$$

The composite Windows 8 rotation matrix using the Microsoft rotation sequence is:

$$\boldsymbol{R} = \boldsymbol{R_y}(\phi)\boldsymbol{R_x}(\theta)\boldsymbol{R_z}(\psi) = \begin{pmatrix} cos\,\phi & 0 & -sin\phi \\ 0 & 1 & 0 \\ sin\phi & 0 & cos\phi \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{pmatrix}\begin{pmatrix} cos\,\psi & sin\psi & 0 \\ -sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{54}$$

$$= \begin{pmatrix} cos\,\phi\,cos\,\psi - sin\phi sin\theta sin\psi & cos\phi sin\psi + cos\,\psi\,sin\phi sin\theta & -sin\phi\,cos\,\theta \\ -cos\,\theta\,sin\psi & cos\,\theta\,cos\,\psi & sin\theta \\ cos\,\psi\,sin\phi + cos\,\phi\,sin\psi sin\theta & sin\phi sin\psi - cos\,\phi\,cos\,\psi\,sin\theta & cos\,\phi\,cos\,\theta \end{pmatrix} \tag{55}$$

4.3 Orientation Quaternion in Terms of Euler Angles

The individual Euler angle rotation quaternions in the Windows 8 coordinate system are:

$$q_x(\theta) = cos\left(\frac{\theta}{2}\right) + \boldsymbol{i}sin\left(\frac{\theta}{2}\right) \tag{56}$$

$$q_y(\phi) = cos\left(\frac{\phi}{2}\right) + \boldsymbol{j}sin\left(\frac{\phi}{2}\right) \tag{57}$$

$$q_z(\psi) = cos\left(\frac{\psi}{2}\right) + \boldsymbol{k}sin\left(\frac{\psi}{2}\right) \tag{58}$$

The Windows 8 rotation quaternion $q_{zxy} = q_z(\psi)q_x(\theta)q_y(\phi)$ evaluates to:

$$q_{zxy} = q_z(\psi)q_x(\theta)q_y(\phi) = \left\{cos\left(\frac{\psi}{2}\right) + \boldsymbol{k}sin\left(\frac{\psi}{2}\right)\right\}\left\{cos\left(\frac{\theta}{2}\right) + \boldsymbol{i}sin\left(\frac{\theta}{2}\right)\right\}\left\{cos\left(\frac{\phi}{2}\right) + \boldsymbol{j}sin\left(\frac{\phi}{2}\right)\right\} \tag{59}$$

$$\begin{aligned} &= \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\} \\ &\quad + \left\{cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{i} \\ &\quad + \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right)\right\}\boldsymbol{j} \\ &\quad + \left\{sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{k} \end{aligned} \tag{60}$$

The elements of the Windows 8 quaternion q_{zxy} are, therefore:

$$q_0 = cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) \quad \textbf{(61)}$$

$$q_1 = cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) \quad \textbf{(62)}$$

$$q_2 = cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) \quad \textbf{(63)}$$

$$q_3 = sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) \quad \textbf{(64)}$$

4.4 Gimbal Lock

Gimbal lock occurs in the Windows 8 coordinate system when the second pitch rotation aligns the first yaw and third roll rotations. This occurs at +90° pitch and –90° pitch angles. The number of degrees of freedom in the Euler angle description of the orientation therefore reduces from 3 to 2 and the roll and yaw angles oscillate with only their sum and difference defined.

At gimbal lock $\theta = 90deg$, equation (55) simplifies to:

$$\boldsymbol{R}_{Win8} = \begin{pmatrix} cos(\psi+\phi) & sin(\psi+\phi) & 0 \\ 0 & 0 & 1 \\ sin(\psi+\phi) & -cos(\psi+\phi) & 0 \end{pmatrix} for\ \theta = 90deg \quad \textbf{(65)}$$

At gimbal lock $\theta = -90deg$, equation (55) simplifies to:

$$\boldsymbol{R}_{Win8} = \begin{pmatrix} cos(\psi-\phi) & sin(\psi-\phi) & 0 \\ 0 & 0 & -1 \\ -sin(\psi-\phi) & cos(\psi-\phi) & 0 \end{pmatrix} for\ \theta = -90deg \quad \textbf{(66)}$$

The orientation matrix and quaternion are stable but only the summed angle $\psi + \phi$ or the differenced angle $\psi - \phi$ can be determined at the two gimbal lock orientations.

4.5 Euler Angle Discontinuities

The Windows 8 rotation matrix $\boldsymbol{R}_y(\phi)\boldsymbol{R}_x(\theta)\boldsymbol{R}_z(\psi)$ is equal to the matrix $\boldsymbol{R}_y(\phi+\pi)\boldsymbol{R}_x(\pi-\theta)\boldsymbol{R}_z(\psi+\pi)$. By direct evaluation:

$$\boldsymbol{R}_y(\phi+\pi)\boldsymbol{R}_x(\pi-\theta)\boldsymbol{R}_z(\psi+\pi) = \begin{pmatrix} cos\,\phi\,cos\,\psi - sin\phi sin\theta sin\psi & cos\phi sin\psi + cos\,\psi\,sin\phi sin\theta & -cos\,\theta\,sin\phi \\ -cos\,\theta\,sin\psi & cos\,\psi\,cos\,\theta & sin\theta \\ cos\,\psi\,sin\phi + cos\,\phi\,sin\psi sin\theta & sin\phi sin\psi - cos\,\phi\,cos\,\psi\,sin\theta & cos\,\phi\,cos\,\theta \end{pmatrix} \quad \textbf{(67)}$$

$$= \begin{pmatrix} cos\,\phi\,cos\,\psi - sin\phi sin\theta sin\psi & cos\phi sin\psi + cos\,\psi\,sin\phi sin\theta & -cos\,\theta\,sin\phi \\ -cos\,\theta\,sin\psi & cos\,\psi\,cos\,\theta & sin\theta \\ cos\,\psi\,sin\phi + cos\,\phi\,sin\psi sin\theta & sin\phi sin\psi - cos\,\phi\,cos\,\psi\,sin\theta & cos\,\phi\,cos\,\theta \end{pmatrix} = \boldsymbol{R}_y(\phi)\boldsymbol{R}_x(\theta)\boldsymbol{R}_z(\psi) \quad \textbf{(68)}$$

Equations (67) and (68) show that there are two solutions for the Windows 8 Euler angles for a given rotation matrix. If ϕ, θ and ψ are one solution for the roll, pitch and yaw angles for a given orientation matrix, then so is the solution given by roll angle 180° plus ϕ, pitch angle 180° minus θ and yaw angle 180° plus ψ.

The Windows 8 specification removes the two solutions by restricting the roll angle θ to the range –90° to +90°.

Microsoft: *"The following ranges are used for the representation of Yaw, Pitch and Roll:*

- *0.0°≤ Yaw < 360.0°*
- *–180.0°≤ Pitch < 180.0°*
- *–90.0°≤ Roll < 90.0°"*

Equations (67) and (68) state that whenever the roll angle is above 90° or less than –90° then the roll angle should increase by 180° (the $\boldsymbol{R}_y(\phi + \pi)$ term), the pitch angle should be negated and 180° added (the $\boldsymbol{R}_x(\pi - \theta)$ term) and the yaw/compass heading should increase by 180° (the $\boldsymbol{R}_z(\psi + \pi)$ term). These discontinuities are shown in Figure 10 through Figure 12 and should be compared with Figure 21 in the Microsoft Inc. specification: *Integrating Motion and Orientation Sensors*.

Figure 2 shows the Euler angles as the PCB is rotated through 360° in roll from the default starting position of being flat and pointed northward. The roll angle reaches 90°, has a discontinuity to –90°, increases to 90° and then has another discontinuity to –90°. The pitch and yaw/compass heading angles suffer 180° discontinuities at 90° and –90° roll angles.

Figure 3 shows the Euler angles as the PCB is rotated 360° in pitch from the same starting position. The pitch angle increases smoothly with the acceptable modulo 360° discontinuity between the equivalent angles of –180° and +180°.

Figure 4 shows the yaw/compass angle behavior as the PCB is rotated 360° in yaw while remaining flat. The yaw/compass heading increases smoothly with the acceptable modulo 360° discontinuity when pointed northward.

The 180° compass heading change when the Windows 8 roll angle passes through –90° or +90° is, like Android, arguably not ideal from an ergonomic standpoint. However, this behavior is forced mathematically onto the Microsoft specification by the unfortunate properties of Euler angles and the requirement to limit the roll angle range from –90° to +90°.

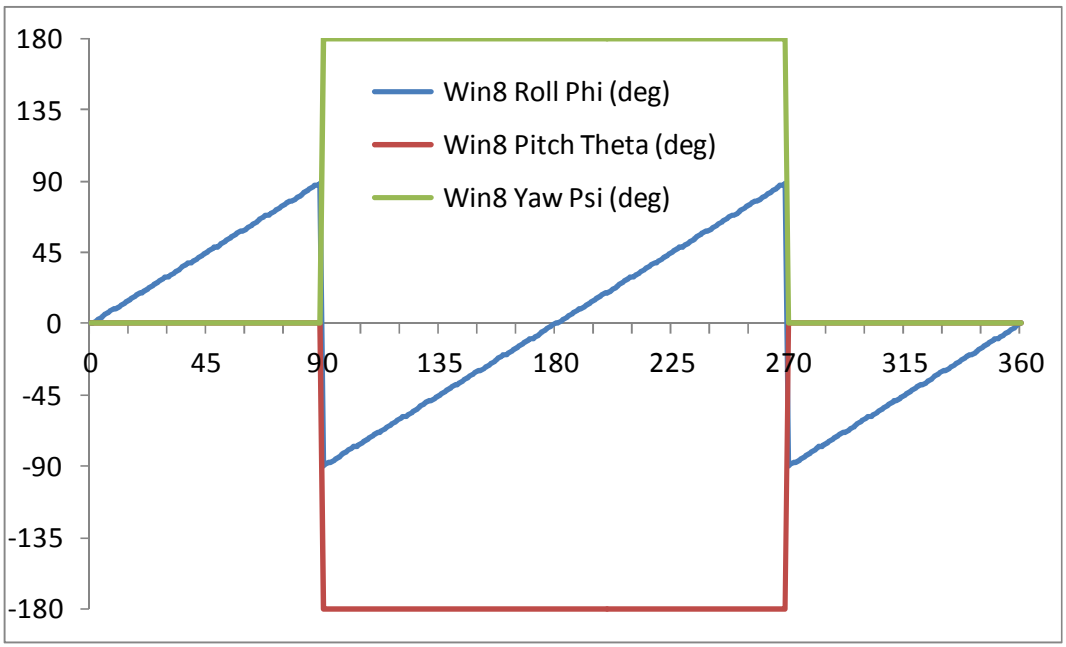

Figure 10. 360° roll rotation at zero pitch and yaw in the Windows 8 coordinate system

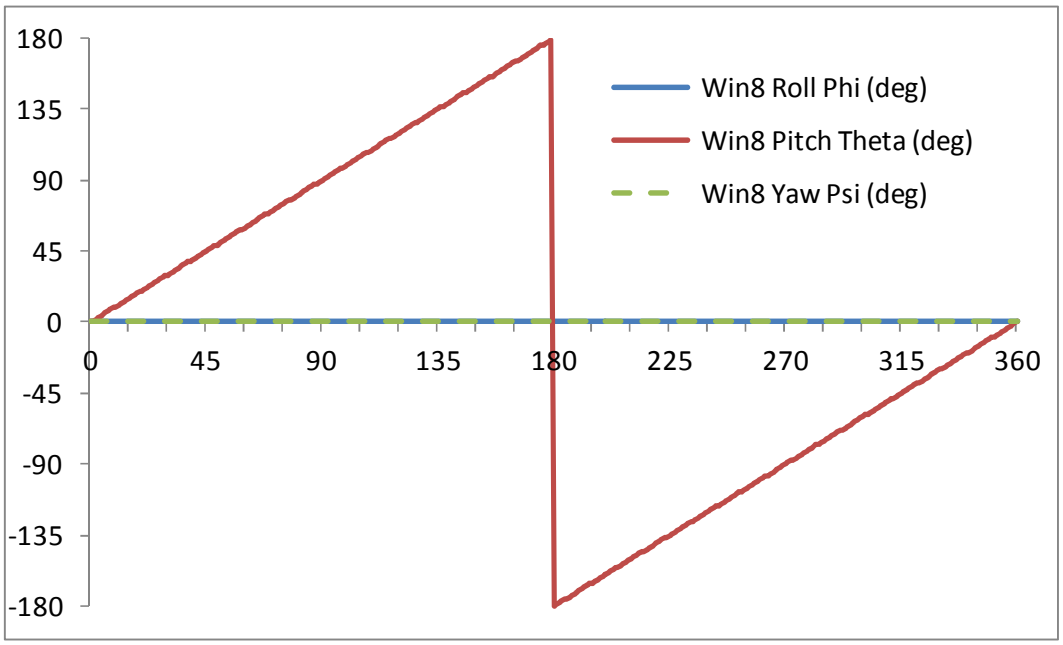

Figure 11. 360° pitch rotation at zero roll and yaw in the Windows 8 coordinate system

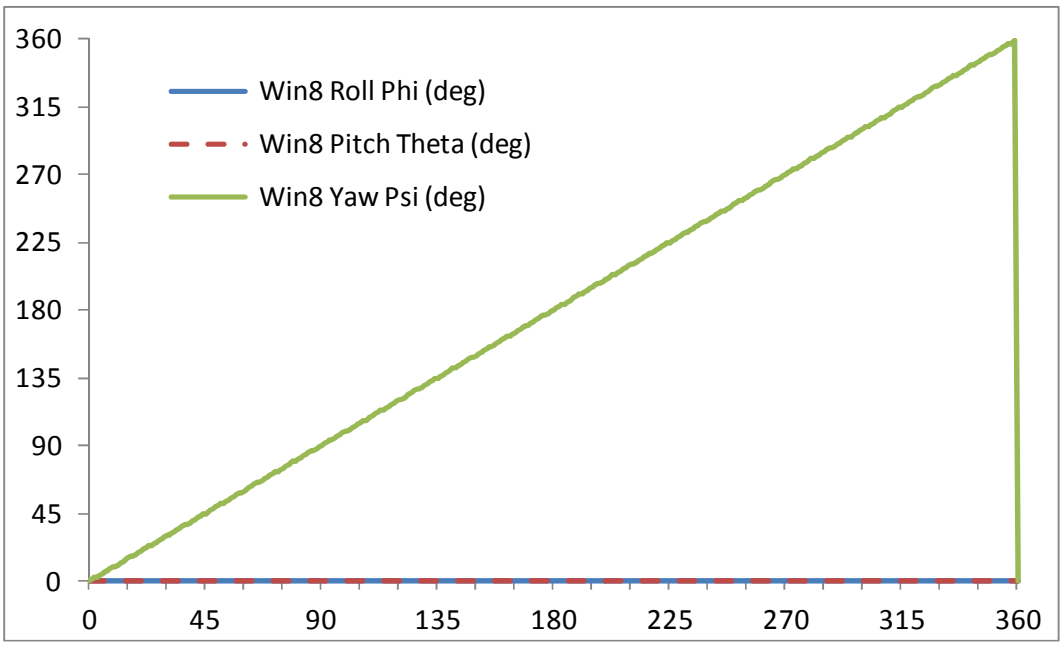

Figure 12. 360° yaw rotation at zero roll and pitch in the Windows 8 coordinate system

4.6 Calculation of Euler Angles from Rotation Matrix

This section documents function `fWin8AnglesDegFromRotationMatrix`, which computes the Euler angles from the Windows 8 rotation matrix as defined in equation (55):

The solution for the roll angle ϕ is:

$$\phi = tan^{-1}\left(\frac{-R_{xz}}{R_{zz}}\right), -90 \leq \phi < 90\ deg \tag{69}$$

The pitch angle θ has range $-180° \leq \theta < 180°$, but is first computed in the range –90° to 90° using:

$$\theta = sin^{-1}\left(R_{yz}\right), -90 \leq \theta < 90\ deg \tag{70}$$

Because ϕ is in the range –90° to 90°, it follows that $cos\,\phi$ is nonnegative and that $R_{zz} = cos\,\phi\,cos\,\theta$ has the same sign as cos θ and can be used to correct θ into the range $-180° \leq \theta < 180°$:

$$\theta \leftarrow \pi - \theta\ if\ R_{zz} < 0\ and\ \theta > 0 \tag{71}$$

$$\theta \leftarrow -\pi - \theta\ if\ R_{zz} < 0\ and\ \theta \leq 0 \tag{72}$$

The general solution for the yaw angle ψ for non-zero cos θ is given by:

$$\psi = tan^{-1}\left(\frac{-cos\,\theta\,R_{yx}}{cos\,\theta\,R_{yy}}\right), 0 \leq \psi < 360\ deg\,, \theta \neq -90deg, \theta \neq 90deg \tag{73}$$

At gimbal lock, equations (65) and (66) give:

$$tan(\psi + \phi) = \left(\frac{R_{xy}}{R_{xx}}\right) for\ \theta = 90deg \tag{74}$$

$$tan(\psi - \phi) = \left(\frac{R_{xy}}{R_{xx}}\right) for\ \theta = -90deg \tag{75}$$

The Windows 8 compass heading angle ρ is the negative (modulo 360°) of the yaw angle ψ:

$$\rho = -\psi \tag{76}$$

The tilt angle from vertical χ can be determined from the scalar product of the rotated gravity vector and the downward z-axis, giving:

$$cos\chi = \left\{\boldsymbol{R}_{Win8}\begin{pmatrix}0\\0\\-1\end{pmatrix}\right\}\cdot\begin{pmatrix}0\\0\\-1\end{pmatrix} = -\begin{pmatrix}R_{xz}\\R_{yz}\\R_{zz}\end{pmatrix}\cdot\begin{pmatrix}0\\0\\-1\end{pmatrix} = R_{zz} = cos\,\theta\,cos\phi \tag{77}$$

4.7 Direction of Windows 8 Rotation Matrix

The Freescale Sensor Fusion Library software defines the orientation matrix as transforming a vector from the global to the sensor coordinate frame for all coordinate systems. The rotation matrices and quaternions listed in the Microsoft Inc. specification: *Integrating Motion and Orientation Sensors* are the transpose and conjugate of the Freescale standard and, therefore, refer to transformation from the sensor frame to the global frame. To convert from one to the other, take the conjugate of the quaternion or transpose the matrix.

5 References

- Android specification available at: *Android APIs>SensorEvent*
- Microsoft Inc. specification: *Integrating Motion and Orientation Sensors*

6 Revision History

Revision history

Rev. No.	Date	Description
1	9/2015	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN5017
Revision 1, 9/2015