life.augmented

DT0104
Design tip

Exploiting the magnetometer as a virtual gyroscope at low and ultra-high spin rates

By Andrea Vitali

Main components	
LSM303AH	Ultra-compact high-performance eCompass module: ultra-low-power 3D accelerometer and 3D magnetometer
LSM303AGR	Ultra-compact high-performance eCompass module: ultra-low-power 3D accelerometer and 3D magnetometer
LIS2MDL	Digital output magnetic sensor: ultra-low-power, high-performance 3-axis magnetometer
LIS3MDL	Digital output magnetic sensor: ultra-low-power, high-performance 3-axis magnetometer

Purpose and benefits

This design tip explains how to exploit magnetometer data to compute angular velocities, effectively emulating a gyroscope.

Benefits:

- Enhanced functionality with respect to data fusion provided by the MotionFX library.
- Reduction of system footprint and/or cost with respect to using a full 9-axis sensor comprising an accelerometer, a magnetometer and a gyroscope.
- Short essential implementation which enables easy customization and enhancement by the end-user. Easy to use on every microcontroller.
- Ultra-high spin rate (18000-180000 dps for magnetometers capable of 100-1000 Hz sampling) well above full scale of gyroscope (2000-4000 dps)

Description of the basic algorithm

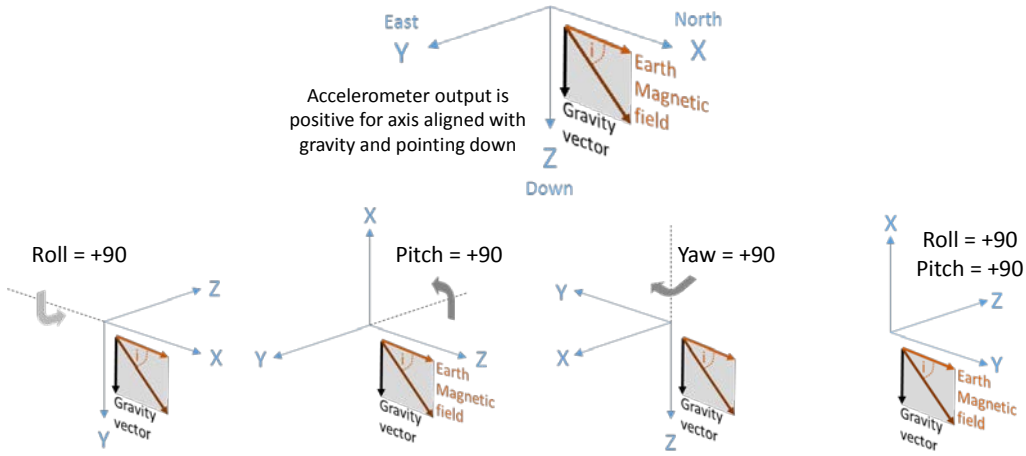
Looking at Figure 1, angular velocities can be computed from the first difference of the angles, which in turn are computed using the a = atan2(y,x) function fed with magnetometer data. Magnetometer data must be free of hard/soft-iron effects.

```
function [wx,wy,wz] = mag2gyro_basic(m,m2)
  mx=m(1); my=m(2); mz=m(3); % new mag data, must be free of hard/soft iron effects
  m2x=m2(1); m2y=m2(2); m2z=m2(3); % old mag data
  ax1 = atan2(my,mz); ax2 = atan2(m2y,m2z); wx = ax1-ax2; % range -2pi..2pi
  ay1 = atan2(mz,mx); ay2 = atan2(m2z,m2x); wy = ay1-ay2;
  az1 = atan2(mx,my); az2 = atan2(m2x,m2y); wz = az1-az2;
  % atan2(y,x) -> L2=x*x+y*y, if L2 near-zero or rapidly varying, discard angle
```

```
  % threshold may be based on earth magnetic field vector modulus
  if(wx>=pi),wx=-2*pi+wx; end; if(wx<=-pi),wx=2*pi+wx; end; % range -pi..+pi
  if(wy>=pi),wy=-2*pi+wy; end; if(wy<=-pi),wy=2*pi+wy; end;
  if(wz>=pi),wz=-2*pi+wz; end; if(wz<=-pi),wz=2*pi+wz; end;
end
```

The output values of the function are the angular velocities of the device, in terms of sensor body coordinates and in degrees per sample. The virtual gyroscope output is computed by multiplying the output of the function (degrees per sample) by the sampling frequency of the sensor (samples per second), so that the unit of measure is degrees per second (dps).

Figure 1. Reference orientation for input data from magnetometer, and reference orientation for output data: roll, pitch and yaw angular velocities



Roll, Pitch, Yaw degrees	Acc (Gravity vector) G = 1g = 1000mg	Mag (Earth mag field) B = 10..90uT = 100..900 mG
0, 0, 0	0, 0, **+G**	**+B Cos(i),** 0, +B Sin(i)
+90, 0, 0	0, **+G**, 0	**+B Cos(i),** +B Sin(i), 0
0, **+90**, 0	**-G**, 0, 0	-B Sin(i), 0, **+B Cos(i)**
0, 0, **+90**	0, 0, **+G**	0, **-B Cos(i),** +B Sin(i)
+90, **+90**, 0	**-G**, 0, 0	-B Sin(i), **+B Cos(i),** 0

Notes:

- The output of the function can be unstable and unreliable. It is recommended to check the length of the vector fed to the atan(y,x) function: **L = sqrt(x*x+y*y)**. If the length is near zero or if it is rapidly varying, then it is better to discard the output. Median filtering can also be used to reject outliers.
- Except for a rotation around the world reference Y-axis, the output will be inaccurate because the Earth's magnetic field vector is not simply rotating in the XZ or YZ body plane. This is the rationale behind the check on the vector length.

- If the rotation axis is the same as the Earth's magnetic field vector, then the output will be zero, because the data output from the magnetometer will be constant (except for the noise).

- On embedded systems, the a = atan2(y,x) function can be efficiently evaluated using the CORDIC algorithm which only uses table lookup, together with binary shift and add operations.

Description of the simplest algorithm (derived from the basic algorithm)

The derivative of the a = atan2(y,x) function can be approximated by **da = (x – y) / (x^2+y^2)**. This formula does not contain any trigonometric function, making it particularly efficient in embedded systems. This is the corresponding function:

```
function [wx,wy,wz] = mag2gyro(m,md)
  mx=m(1); my=m(2); mz=m(3); % mag data, must be free of hard/soft iron effects
  mxd=md(1); myd=md(2); mzd=md(3); % mag data derivative
  mx2=mx*mx; my2=my*my; mz2=mz*mz;
  wx=(myd*mz - my*mzd)/(my2+mz2); % if denominators are near-zero or rapidly varying
  wy=(mzd*mx - mz*mxd)/(mx2+mz2); % then discard output; threshold may be based on
  wz=(mxd*my - mx*myd)/(mx2+my2); % earth magnetic field modulus
end
```

The virtual gyroscope output is computed by multiplying the output of the function (degrees per sample) by the sampling frequency of the sensor (samples per second), so that the unit of measure is degrees per second (dps).

Notes:

- The output of the function can be unstable and unreliable. It is recommended to check the magnitude of the denominators. If the magnitude is near zero or if it is rapidly varying, then it is better to discard the output. Median filtering can also be used to reject outliers.

- The output of the function can be very **noisy** and it can also be **attenuated** depending on how the magnetometer data derivative is computed using finite differences (see the paragraph "Notes on the computation of the derivative").

- As for the basic algorithm, except for a rotation around the world reference Y-axis, the output will be inaccurate. Also, if the rotation axis is the same as the Earth's magnetic field vector, then the output will be zero. (see Notes in the previous paragraph).

Description of the best algorithm

When the device is rotating by a given angle around a given rotation axis, the Earth's magnetic field vector sweeps a circle on the sphere. The plane corresponding to the circle can be computed by fitting at least three magnetometer data points. The center of the circle is then computed by projection of the origin on the plane. The normal to the plane is the rotation axis. The angular velocity is computed by looking at the vector going from the center of the circle to the point of the circumference identified by the Earth's magnetic field.

```
function [ax,w,Qupd] = mag2ax_w_Qupd(m,m2,m3)
```

```
  % m is the newest, m3 is the oldest magnetometer data point, 1x3 row vectors
  [n,p]=planefit([m;m2;m3]); % get normal and point in the plane
  p0=(p(1)*n(1)+p(2)*n(2)+p(3)*n(3))*n; % -dot(-p,n)*n, center of rotation
  U=m-p0; V=m2-p0; % vectors sweeping the circle
  UdotV = U(1)*V(1)+U(2)*V(2)+U(3)*V(3);
  UvectV(1) = U(2)*V(3)-U(3)*V(2);
  UvectV(2) = U(3)*V(1)-U(1)*V(3);
  UvectV(3) = U(1)*V(2)-U(2)*V(1);
  UvectVn = sqrt(UvectV(1)*UvectV(1)+UvectV(2)*UvectV(2)+UvectV(3)*UvectV(3));
  w=atan2(UvectVn,UdotV); % angular velocity
  % L2=UvectVn*UvectVn+UdotV*UdotV, if L2 near-zero or rapidly varying, discard w
  ax=[0,0,0]; if UvectVn>0, ax=UvectV/UvectVn; end; % rotation axis
  QC=cos(w/2); QS=sin(w/2);
  Qupd = [QC,QS*ax(1),QS*ax(2),QS*ax(3)]; % quaternion update
end
```

For plane fitting, one of these functions can be used:

```
function [n,p]=planefitls(xyz) % xyz: Nx3 matrix, each line is a data point
  % least square solution, minimize distance along main axis
  p=mean(xyz,1); % p point in the plane
  x=xyz(:,1)-p(1); y=xyz(:,2)-p(2); z=xyz(:,3)-p(3);
  xx=sum(x.*x); yy=sum(y.*y); zz=sum(z.*z);
  xy=sum(x.*y); xz=sum(x.*z); yz=sum(y.*z);
  Dx=yy*zz-yz*yz; Dy=xx*zz-xz*xz; Dz=xx*yy-xy*xy; % use largest
                  D=abs(Dx); flag=0;
  if abs(Dy)>D, D=abs(Dy); flag=1; end;
  if abs(Dz)>D, D=abs(Dz); flag=2; end;
  if D==0, n=[0,0,0]; return; end;
  switch flag
    case 0, a=1;                b=(xz*yz-xy*zz)/Dx; c=(xy*yz-xz*yy)/Dx;
    case 1, a=(yz*xz-xy*zz)/Dy ; b=1;               c=(xy*xz-yz*xx)/Dy;
    case 2, a=(yz*xy-xz*yy)/Dz;  b=(xz*xy-yz*xx)/Dz; c=1;
  end
  n=[a,b,c]./sqrt(a*a+b*b+c*c); % normal to plane
end

function [n,p,B]=planefiteig(xyz) % xyz: Nx3 matrix, each line is a data point
  % least square solution, minimize orthonormal distance
  p=mean(xyz,1); % point in the plane
  xyz(:,1)=xyz(:,1)-p(1); xyz(:,2)=xyz(:,2)-p(2); xyz(:,3)=xyz(:,3)-p(3);
  [eigvect,eigval]=eig(xyz' * xyz); % eigenvectors of a 3x3 matrix
  n=eigvect(:,1)'; % normal to plane
  B=eigvect(:,2:end)'; % rows are an orthonormal basis for the plane
end
```

Once the rotation axis and the angular velocity are known, the current orientation can be updated. From the current quaternion and its derivative, the angular velocities in the sensor body coordinates can be computed. The output is **attenuated at high spin rates** (-33% at 180 degrees/sample).

The current quaternion can also be converted to Euler angles representation. From current Euler angles and their derivatives, the angular velocities in sensor body coordinates can be computed. Output is not attenuated at high spin rates. However the derivative and the output are **not correct at the North and South pole singularities** (pitch = ±90 degrees).

The corresponding reference code is the following and includes references to other algorithms for comparison purposes:

```
% mag data must be free of hard/soft iron effects
m3=m2; m2=m; m=MagData(i,:); % new mag data point, keep previous, 1x3 vectors

% basic algo: first difference of atan2
[wx,wy,wz]=mag2gyro_basic(m,m2); % multiply by sampling frequency Fs
```

```
% simplest algo: atan2 derivative
md=m-m2; % mag data derivative, attenuation at high spin rates
[wx,wy,wz]=mag2gyro(m,md); % multiply by sampling frequency Fs

% best algo: axis and angular speed of rotation
[ax,w,Qupd]=mag2ax_w_Qupd(m,m2,m3); % multiply w by sampling frequency Fs

Q2=Q; Q=quatbyquat(Q,Qupd); % current quat, keep previous
QD=quat2diff(Q,Q2); % quat derivative, attenuation at high spin rates
[wx,wy,wz]=quat2gyro_body(Q,QD); % multiply by sampling frequency Fs

R2=R; P2=P; Y2=Y; [R,P,Y]=quat2euler(Q); % current Euler from quat, keep previous
[RD,PD,YD]=euler2diff(R,P,Y,R2,P2,Y2); % singularities: not correct when P==+/-pi/2
[wx,wy,wz]=euler2gyro(R,P,Y,RD,PD,YD); % multiply by sampling frequency Fs
```

The virtual gyroscope output is computed by multiplying the output of the function (degrees per sample) by the sampling frequency of the sensor (samples per second), so that the unit of measure is degrees per second (dps).

For quaternion product, quaternion to Euler angles conversion, backward differences, and computation of angular velocities, the following functions can be used:

```
function R = quatbyquat(P,Q)
  pw=P(1); px=P(2); py=P(3); pz=P(4);
  qw=Q(1); qx=Q(2); qy=Q(3); qz=Q(4);
  rw = pw*qw - px*qx - py*qy - pz*qz;
  rx = pw*qx + px*qw + py*qz - pz*qy;
  ry = pw*qy - px*qz + py*qw + pz*qx;
  rz = pw*qz + px*qy - py*qx + pz*qw;
  R = [rw rx ry rz];
end

function Qdelta = quat2diff(Qnew,Qold)
  q1w=Qnew(1); q1x=Qnew(2); q1y=Qnew(3); q1z=Qnew(4);
  q0w=Qold(1); q0x=Qold(2); q0y=Qold(3); q0z=Qold(4);
  if (q1w*q0w + q1x*q0x + q1y*q0y + q1z*q0z)<0, % dot product
    q0w=-q0w; q0x=-q0x; q0y=-q0y; q0z=-q0z;
  end;
  qdw=q1w-q0w; qdx=q1x-q0x; qdy=q1y-q0y; qdz=q1z-q0z; % diff
  Qdelta = [qdw,qdx,qdy,qdz];
end

function [phi,theta,psi]=quat2euler(Q)
  % North-East-Down reference frame
  % Roll(phi) Pitch(theta) Yaw(psi), angles in radians
  if(Q(1)<0), Q=-Q; end; % psi range -pi..+pi at North & South
  qw=Q(1); qx=Q(2); qy=Q(3); qz=Q(4);
  qw2=qw*qw; qx2=qx*qx; qy2=qy*qy; qz2=qz*qz;
  m=qw2+qx2+qy2+qz2; th=0.5*m-0.01; t=qw*qy-qz*qx; % 0.01 tolerance for singularity
  if(m<=0.0),  psi=0.0;              theta=0.0;   phi=0.0; return; end; % error
  if(t>=+th),  psi=-2.0*atan2(qx,qw); theta=+pi/2; phi=0.0; return; end; % North
  if(t<=-th),  psi=+2.0*atan2(qx,qw); theta=-pi/2; phi=0.0; return; end; % South
  psi=atan2(2.0*(qw*qz+qx*qy),qw2+qx2-qy2-qz2);
  t=2.0*t/m; if(abs(t)>1.0), theta=0.0; else theta=asin(t); end;
  phi=atan2(2.0*(qw*qx+qy*qz),qw2-qx2-qy2+qz2);
end

function [RD,PD,YD] = euler2diff(R1,P1,Y1,R0,P0,Y0) % 1:new, 0:old
  % when Pitch=+90 deg -> Yaw-Roll, when Pitch=-90 deg -> Yaw+Roll
  RD=mod(R1-R0,2*pi); if RD>=pi, RD=-2*pi+RD; end; % range 0..2pi to -pi..+pi
  PD=mod(P1-P0,2*pi); if PD>=pi, PD=-2*pi+PD; end;
  YD=mod(Y1-Y0,2*pi); if YD>=pi, YD=-2*pi+YD; end;
end

% angular velocity in sensor body coordinates = 2 * conj(q) * q'
function [wx,wy,wz] = quat2gyro_body(Q,QD)
  pw=Q(1);  px=Q(2); py=Q(3); pz=Q(4); % quat
  qw=QD(1); qx=QD(2); qy=QD(3); qz=QD(4); % derivative
```

```
  wx = 2*(-qw*px + qx*pw + qy*pz - qz*py );
  wy = 2*(-qw*py - qx*pz + qy*pw + qz*px );
  wz = 2*(-qw*pz + qx*py - qy*px + qz*pw );
  % reduce to range -2pi..+2pi and then to -pi..+pi
  wx=rem(wx,2*pi); if(wx>=+pi),wx=-2*pi+wx; end; if(wx<=-pi),wx=+2*pi+wx; end;
  wy=rem(wy,2*pi); if(wy>=+pi),wy=-2*pi+wy; end; if(wy<=-pi),wy=+2*pi+wy; end;
  wz=rem(wz,2*pi); if(wz>=+pi),wz=-2*pi+wz; end; if(wz<=-pi),wz=+2*pi+wz; end;
end

%  angular velocity in world reference coordinates = 2 * q' * conj(q)
function [wx,wy,wz] = quat2gyro_world(Q,QD)
  pw=Q(1);  px=Q(2); py=Q(3); pz=Q(4); % quat
  qw=QD(1); qx=QD(2); qy=QD(3); qz=QD(4); % derivative
  wx = 2*(-qw*px + qx*pw - qy*pz + qz*py );
  wy = 2*(-qw*py + qx*pz + qy*pw - qz*px );
  wz = 2*(-qw*pz - qx*py + qy*px + qz*pw );
  % reduce to range -2pi..+2pi and then to -pi..+pi
  wx=rem(wx,2*pi); if(wx>=+pi),wx=-2*pi+wx; end; if(wx<=-pi),wx=+2*pi+wx; end;
  wy=rem(wy,2*pi); if(wy>=+pi),wy=-2*pi+wy; end; if(wy<=-pi),wy=+2*pi+wy; end;
  wz=rem(wz,2*pi); if(wz>=+pi),wz=-2*pi+wz; end; if(wz<=-pi),wz=+2*pi+wz; end;
end

function [wx,wy,wz] = euler2gyro(phi,theta,psi,phid,thetad,psid)
  % angles are modulo 2pi: from -pi to +pi, diff can have false discontinuities
  if(phid  >pi),phid  =-2*pi+phid;   end; if(phid  <-pi),phid  =2*pi+phid;   end;
  if(thetad>pi),thetad=-2*pi+thetad; end; if(thetad<-pi),thetad=2*pi+thetad; end;
  if(psid  >pi),psid  =-2*pi+psid;   end; if(psid  <-pi),psid  =2*pi+psid;   end;
  wx = phid                - psid*sin(theta);
  wy =      thetad*cos(phi) + psid*sin(phi)*cos(theta);
  wz =     -thetad*sin(phi) + psid*cos(phi)*cos(theta);
  % reduce to range -2pi..+2pi and then to -pi..+pi
  wx=rem(wx,2*pi); if(wx>=+pi),wx=-2*pi+wx; end; if(wx<=-pi),wx=+2*pi+wx; end;
  wy=rem(wy,2*pi); if(wy>=+pi),wy=-2*pi+wy; end; if(wy<=-pi),wy=+2*pi+wy; end;
  wz=rem(wz,2*pi); if(wz>=+pi),wz=-2*pi+wz; end; if(wz<=-pi),wz=+2*pi+wz; end;
end
```

Description of the simplest algorithm for ultra-high spin rate

The algorithms presented can estimate the angular velocity in the range of ±180 degrees per sample. This means that if the magnetometer can sample at 100 Hz, the maximum angular velocity will be 18000 dps. If it can sample at 1000 Hz, the maximum angular velocity will be 180000 dps.

If the rotation axis is constant and one is only interested in the **spin rate** around the rotation axis, the algorithm can be greatly simplified as the magnetometer data will follow a sinusoidal pattern, where one cycle corresponds to one full revolution. Cycles and revolutions can be counted in one of two ways:

- by identifying the local minima or the local maxima, or
- by counting zero-crossings after the average value has been subtracted.

If at least two of the magnetometer axes will exhibit the pattern, it is then possible to validate the counter by cross-checking. The only exception is when the rotation axis is coincident with the Earth's magnetic field vector, in which case no change is visible in the data. If R revolutions are identified in N samples, and if the sampling frequency is Fs, then it is possible to compute the revolutions per minute (**rpm = 60*R*Fs/N**) or the angular velocity in degrees per second (**dps = 60*rpm = 360*R*Fs/N**).

Notes on the computation of the derivative

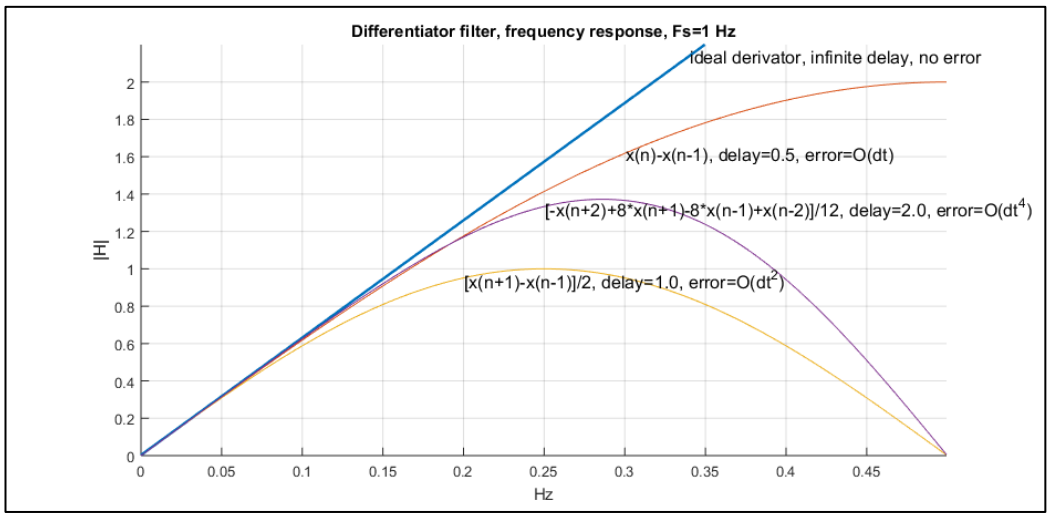
The derivative of a signal can be approximated by a digital difference.

- The simplest formula is based on subtraction of the previous sample: $\mathbf{x'=x_n-x_{n-1}}$, this is known as backward difference. The disadvantages are that the delay is fractional (0.5 samples) and, most importantly, that high-frequency noise is amplified.

- A more accurate formula is the central difference: $\mathbf{x'= (x_{n+1} - x_{n-1})/2}$. The advantages are that the delay is the smallest integer (1 sample) and, most importantly, that high-frequency noise is attenuated; however low frequencies are also attenuated.

- A very accurate formula is the following: $\mathbf{x' = (-x_{n+2} +8x_{n-1} -8x_{n+1} +x_{n-2})/12}$. The advantages are that the delay is an integer (2 samples), high-frequency noise is still attenuated, while a larger range of low frequencies is kept. See the figure below for a comparison.

Figure 2. Frequency response for ideal derivator



Generally speaking, it is recommended to smooth the signal before any algorithm is applied. The smooth derivative can also be computed directly by finite-impulse-response (FIR) filters. Aforementioned formulas to compute the digital difference are in fact FIR filters with 2, 3 and 5 coefficients respectively: [1 -1], [1/2, 0, -1/2], [-1/12, 2/3, -2/3, 1/12]. To compute the smooth signal and its derivatives, it is recommended to look at **Savitzy-Golay** filters.

Support material

Related design support material
BlueMicrosystem1, Bluetooth low energy and sensors software expansion for STM32Cube
Open.MEMS, MotionFX, Real-time motion-sensor data fusion software expansion for STM32Cube
Documentation
Design Tip, DT0058, Computing tilt measurement and tilt-compensated eCompass
Design Tip, DT0059, Ellipsoid or sphere fitting for sensor calibration
Design Tip, DT0060, Exploiting the gyroscope to update tilt measure and eCompass

Revision history

Date	Version	Changes
28-Aug-2018	1	Initial release

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.