# **Pulse Guide: Methodology**

Pulse Guiding is a method of keeping the telescope pointing towards a defined star in the field of vision of a Spotter scope. A CDD camera image of the sky pointed to by the Spotter scope keeps track of the position of a defined star in its image. As the star appears to move (due to the Earth’s rotation) the movement will be detected by PHD, PHD commands will be sent to the mount to move the telescope to counter this apparent movement, the commands from ASCOM (PHD) (PulseGuide) will contain a direction and a duration, the speed already having been defined using the GuideRateAscension and GuideRateDeclination properties, a double containing the speed, which will be a user (via the ASCOM .dll) defined fraction of SIDEREAL rate to be added to the current tracking speed for the duration (in milliseconds) of the PulseGuide command.

The ASCOM command is of the format :

void PulseGuide(

GuideDirections *Direction*,

int *Duration*

)

where Direction is given by:

0 = North (+ declination/altitude).

1 = South (- declination/altitude).

2 = East (+ right ascension/azimuth).

3 = West (- right ascension/azimuth)

and Duration is in milliseconds.

The rate of motion for movements about the right ascension axis is specified by the [GuideRateRightAscension](http://www.ascom-standards.org/Help/Platform/html/P_ASCOM_DeviceInterface_ITelescopeV3_GuideRateRightAscension.htm) property. The rate of motion for movements about the declination axis is specified by the [GuideRateDeclination](http://www.ascom-standards.org/Help/Platform/html/P_ASCOM_DeviceInterface_ITelescopeV3_GuideRateDeclination.htm) property.

Upon receipt of a PulseGuide command at the HUB, the HUB will:

* 1. Determine the motor (field[0]) to which the Pulse Guide needs to be applied:
     1. 0 and 1 = altitude motor
     2. 2 and 3 = azimuth motor
  2. Send the command to the appropriate motor:

int PulseGuide\_f(String command) { // f12}

Parse\_Command\_to\_Fieldf(command);

switch ((int) command\_fieldf[0]) {

case North:

Send\_Message\_to\_Driver(Altitude\_motor, PulseGuide\_command, command\_fieldf[0], command\_fieldf[1], 0, 0); // send message to driver

break;

case South:

Send\_Message\_to\_Driver(Altitude\_motor, PulseGuide\_command, command\_fieldf[0], command\_fieldf[1], 0, 0); // send message to driver

break;

case East:

Send\_Message\_to\_Driver(Azimuth\_motor, PulseGuide\_command, command\_fieldf[0], command\_fieldf[1], 0, 0); // send message to driver

break;

case West:

Send\_Message\_to\_Driver(Azimuth\_motor, PulseGuide\_command, command\_fieldf[0], command\_fieldf[1], 0, 0); // send message to driver

break;

}

Upon receipt of the modified PulseGuide command the addressed motor will:

1. Save the appropriate variables:
   1. PulseGuide\_Direction = Drivermessage.bearing
   2. PulseGuide\_Duration = Drivermessage.field2
   3. Previous\_Motor\_Speed = Motordrive.period
2. Calculate the new speed:
   1. If PulseGuide\_Direction = 0 speed = Motordrive.period + GuideRateDeclination
   2. If PulseGuide\_Direction = 1 speed = Motordrive.period - GuideRateDeclination
   3. If PulseGuide\_Direction = 2 speed = Motordrive.period + GuideRateRightAscension
   4. If PulseGuide\_Direction = 3 speed = Motordrive.period - GuideRateRightAscension
3. Calculate the new direction:
   1. If PulseGuide\_Direction = 0 (North) direction = CW (Altitude motor)
   2. If PulseGuide\_Direction = 1 (South) direction = ACW (Altitude motor)
   3. If PulseGuide\_Direction = 2 (West) direction = CW (Azimuth motor)
   4. If PulseGuide\_Direction = 3 (East) direction = ACW (Azimuth motor)
4. Turn the Motor On:
   1. Turn\_Motor\_On\_Guiding(speed, direction) {

Set\_Motor\_Direction(direction);

Motordrive.period = speed;

Motordrive.pulse\_rising = Motordrive.period >> 1; // OCR - signal goes high

Motordrive.pulse\_falling = Motordrive.period; // OCRS - signal goes low

InitTimer2();

OC1CONbits.OCM = 0b100; // pin starts low, driven high on OC1P/OC1PS

T2CONbits.TON = 0x01; // Start Timer2

millispulseguiding = 0; // zero the pulseguiding timer

MotorStatus.motorstatus.change = true;

}

* 1. MotorStatus.Motorstatus.pulseguiding = true;

case PulseGuide\_command:

PulseGuide\_Direction = (int) CommandMessage.Drivermessage.bearing; // the required duration, in milliseconds for the Pulse Guide

PulseGuide\_Duration = (long) CommandMessage.Drivermessage.field2; // the direction required for the Pulse Guide, 1 = CW, 0 = ACW

Previous\_Motor\_Speed = Motordrive.period; // save the current direction

MotorStatus.Motorstatus.pulseguiding = true;

millispulseguiding = (long) 0; // zero the pulse guiding timer

switch (PulseGuide\_Direction) {

case NORTH:

Turn\_Motor\_On\_Guiding(Previous\_Motor\_Speed + GuideRateDeclination);

break;

case SOUTH:

Turn\_Motor\_On\_Guiding(Previous\_Motor\_Speed - GuideRateDeclination);

break;

case WEST:

Turn\_Motor\_On\_Guiding(Previous\_Motor\_Speed + GuideRateRightAscension);

break;

case EAST:

Turn\_Motor\_On\_Guiding(Previous\_Motor\_Speed - GuideRateRightAscension);

break;

}

Send\_Reply\_to\_HUB(PulseGuide\_command, good, PulseGuide\_Direction);

break;

1. In the main loop:
   1. If pulseguiding is true
      1. Check if millispulseguiding >= PulseGuideDuration
         1. Motordrive.period = Previous\_Motor\_Speed
         2. Motordrive.dir = Previous\_Motor\_Direction
         3. Turn\_Motor\_On\_Guiding(Motordrive.period, Motordrive.dir)
         4. MotorStatus.motorstatus.pulseguiding = false

if (MotorStatus.Motorstatus.pulseguiding == true) {

if (millispulseguiding >= PulseGuide\_Duration) { // has the specified pulse guiding period expired?

Turn\_Motor\_On\_Guiding(Previous\_Motor\_Speed); // by reloading the saved speed

}

MotorStatus.Motorstatus.pulseguiding = false;

}