Open Drive

an Open Source Implementation
of a
Telescope Mount Controller

Authors:

starrookie

...<others to be added>

Table of Contents

1 Use cases

1.1.1 Use Cases by Priority and SW Version

Use Case	Priority	Version	Remark
1.1.2.5.1 Retrieve Mount Parameters from firmware	0	1	
1.1.2.5.2 Retrieve Mount Control Parameters from firmware			
1.1.3.1.1 Drive RA Motor at constant Speed	0	1	
1.1.3.2.1 Switch to alternate RA speed by handbox switch	1	1	
1.1.3.2.2 Switch to alternate DEC speed by handbox switch	1	1	
1.1.2.1.3 Accept guiding commands from handbox keystrokes	1	1	
1.1.2.1.4 Accept guiding commands from ST4 interface	2	1	

Table 1: Use Cases by priority and version

1.1.2 User Interface

1.1.2.1 TelescopeControl interfaces

1.1.2.1.1 Accept commands from console keyboard

1.1.2.1.2 Accept commands from Web-Server Applicationh

1.1.2.1.3 Accept guiding commands from handbox keystrokes

The user in additon to the RA and DEC motors connects a 4 switch handbox If one or multiple of the push switches are pressed, movement in RA or DEC direction is performed at a constant speed forward or backward depending on the set of switches pressed. If both swicthes referring to RA or DEC direction are pressed, these pressed swicthes are ignored. If no switch is pressed, the mount moves the telescope in RA and DEC at the predefined speeds.

1.1.2.1.4 Accept guiding commands from ST4 interface

1.1.2.1.5 Accept RA/DEC guiding commands from encoder devices

1.1.2.1.6 Accept LX200 commands sent via RS232

1.1.2.1.7 Accept LX200 commands sent via USB

1.1.2.1.8 Display Telescope Control status via console

1.1.2.1.9 Display Telescope Control status via Web Server

1.1.2.2 Provide User Guidance

1.1.2.2.1 Manually Adjust Mount Position from guiding signals by Scheiner Method

1.1.2.2.2 Adjust Mount Position from 3 star mis alignment

1.1.2.3 Telescope Control Commands

1.1.2.3.1 Select Object to follow from object database

1.1.2.3.2 Select Object to follow from RA/DEC coordinates

1.1.2.3.3 Select Object to follow from Alt/AZ coordinates

1.1.2.4 Object Database Commands

1.1.2.4.1 Display Object data from database

1.1.2.4.2 Add Object data to object database from PC File

1.1.2.4.3 Replace Object database by PC-File

1.1.2.4.4 Add Object data to object database from Network connection

1.1.2.4.5 Replace Object database by Network Connection

1.1.2.4.6 Add Object data to Object database data from guiding signals

1.1.2.5 Parameter settings

1.1.2.5.1 Retrieve Mount Parameters from firmware

The user of the mount compiles a version of the OpenDrive Firmware. At compile time he specifies the parameters of his telescope mount, motors and transmission

1.1.2.5.2 Retrieve Mount Control Parameters from firmware

The user of the mount compiles a version of the OpenDrive Firmware. At compile time he specifies the parameters of the motor controller.

1.1.2.5.3 Enter Mount Parameters from console

1.1.2.5.4 Enter Mount Position and Local Time from console

1.1.2.5.5 Receive Mount Position and Local Time from GPS

1.1.2.5.6 Align Telescope Position by 1 star method (Mount position known)

1.1.2.5.7 Align Telescope Position by 2 star method

1.1.2.5.8 Align Telescope Position by 3 star method

1.1.2.5.9 Adjust Mount Position from 3 star misalignment – Alt/Az Motors

1.1.2.5.10 Adjust Mount Position from guiding signals by Scheiner Method– Alt/Az Motors

1.1.2.5.11 Learn PEC from guiding signals

1.1.2.5.12 Learn atmospheric diffraction from guiding signals

1.1.2.5.13 Learn Image Rotator speed from guiding signals according to Scheiner Method

1.1.2.5.14 Store Motor Focus Positions

1.1.2.5.15 Store Image Rotator Speed

1.1.3 Telescope Control

1.1.3.1 Follow Object by Moving in RA/DEC

1.1.3.1.1 Drive RA Motor at constant Speed

The user connects a RA-motor to the OpenDrive Telescope Controller. The motor is moving the telescope at contant speed in RA.

1.1.3.1.2 Drive Telecope to Alt/Az position at speed 0

1.1.3.1.3 Drive Telescope to RA/DEC position at constant speed

1.1.3.1.4 Follow arbitrary position/speed track.

1.1.3.2 Accept guiding corrections while following object

1.1.3.2.1 Switch to alternate RA speed by handbox switch

The user in additon to the RA motor connects a handbox (at least a push switch). If the Push switch is pressed, the movement in RA direction is performed at a different speed.

1.1.3.2.2 Switch to alternate DEC speed by handbox switch

The user connects a DEC motor and a a handbox (at least a second push switch). If the Push switch is pressed, a movement in DEC direction is performed at constant speed.

1.1.3.2.3 Adjust RA Position by Increment/Decrement

1.1.3.2.4 Adjust DEC Position by Increment/Decrement

1.1.3.2.5 Adjust RA Speed by PEC

1.1.3.2.6 Adjust RA/DEC speed for atmospheric diffraction based on Alt/Az position

1.2 Supplemental Motor Control

1.2.1.1 Follow Object by Moving Image Rotator

1.2.1.1.1 Move Image Rotator at constant speed

1.2.1.1.2 Adjust Image Rotator Position by Increment/Decrement

1.2.1.1.3 Adjust Image Rotator Speed by Increment/Decrement

1.2.1.2 Control Motor Focus Driver

1.2.1.2.1 Drive Motor Focus to Position

1.2.1.2.2 Adjust Motor Focus Position by Increment/Decrement

1.2.1.3 Control Mount Altitude/Azimut Motors

1.2.1.3.1 Adjust Mount Altitude Position by Increment/Decrement

1.2.1.3.2 Adjust Mount Azimut Position by Increment/Decrement

2 Architecture Overview

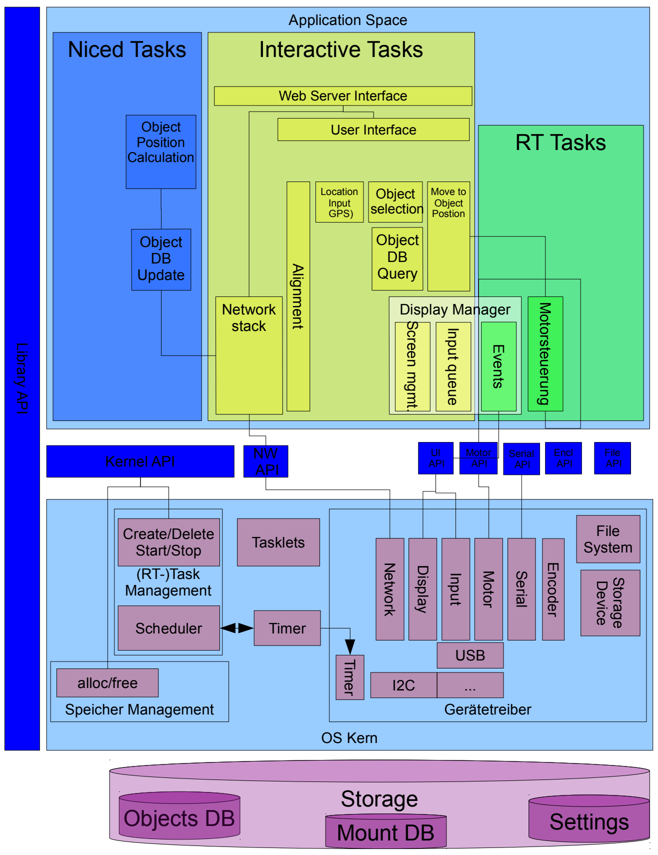

Illustration 1: Block Diagram of Architecture

3 Functional Requirements and Design

3.1 Style Guidelines

3.1.1 This document

3.1.1.1 Functional Specifications

Input Modifiers: Data, which has influence on processing of the function

Parameter Name	Parameter Type	Comment
Direct Parameters		Modifiers passed to the function as call parameters
Indirect Parameters		Modifiers contained in data structures not passed to the function as call parameters
Child Modifiers		Modifiers which influence the behavior of called functions

Table 2: Legend for Input Modifiers

Output Modifiers: Data, which have been influenced by processing the function

Parameter Name	Parameter Type	Comment
Direct Parameters		Modifiers returned from the function as return code
Indirect Parameters		Modifiers contained in data structures not returned by the function as return code
Child Modifiers		Modifiers which have been influenced by called functions

Table 3: Legend for Output Modifiers

3.2 Commonalities

3.2.1 Constant Definitions

Constant	Value	Comment
MAX_MUSTEPS	64	
MAX_MOTORS	8	RA DEC MotorFocus Image Rotator Base Altitude Base Azimut Unassigned Unassigned

3.2.2 Derivative data types

Derivative Type	Basic Type	Comment
task_name	char[16]	
task_function	int (*fn)(int, char *)	Returns return_code
return_code	Int	>=0 on success
task_id	int	

task_state	Enum {
	TASK_NEW,
	TASK_INIT,
	TASK_STOPPED
	TASK_RUNNABLE,
	TASK_SLEEPING,
	TASK_ENDED
	TASK_DESTROYED
	}	

task_struct	Struct {
	int Id
	int argc,
	int refct
	task_list waitlist
	char * argv[];
	}	

task_list	Struct {
	task.waitlist next
	task.waitlist prev
	}	

time_spec	Struct {
	long secs
	long usecs
	}	

phase_spec	int	0..4 * MAX_MUSTEPS-1

Derivative Type	Basic Type	Comment
location_spec	Int	Ticks
speed_spec	Int	Ticks per second
accel_spec	int	Tics per second^2

position_spec	Struct {
	location_spec location
	speed_spec speed
	accel_spec acceleration
	}	

motor_state	Struct {
	boolean mustep;
	time_spec nextUpdate
	position_data current
	position_data target
	}	

motor_driver_spec	Struct {
	tick_function tick
	setspeed_function setspeed
	setlocation_function setlocation
	setposition_function setposition
	}
		Implemented or NULL

wait_event	Struct {
	task_list waiters
	}	

Table 4: Derivative data types

3.2.3 Function Patterns

3.2.3.1 task_function

Parameter Name	Parameter Type	Comment
argc	Int	Number of parameters
argv	*char[argc]	Array of Strings Dimension: argc

Table 5: Input Modifiers for task_function

Parameter Name	Parameter Type	Comment
ReturnCode	return_code	

Table 6: Output Modifiers for task_function

3.3 Core Kernel

3.3.1 Task Management

3.3.1.1 create_task

Parameter Name	Parameter Type	Comment
Name	task_name	
Function	task_function	
Argc	int	
Argv	*Char []	Allocated by Crator

Table 7: Input Modifiers for create_task

Parameter Name	Parameter Type	Comment
ReturnCode	return_code	TaskID or error code
Task	task_struct	Contains copies of argv

Table 8: Output Modifiers for create:task

Functional Specification:

3.3.1.2 modify_task

Parameter Name	Parameter Type	Comment
Task	task_id	
State	task_state	

Table 9: Input Modifiers for modify_task

Parameter Name	Parameter Type	Comment
Task	task_struct	

Table 10: Output Modifiers for modify:task

Functional Specification:

3.3.1.3 schedule

Parameter Name	Parameter Type	Comment
Time	time_spec	
Absolute	int	1: Absolute timeval 0: relative timeval
Niceness	Int	<0 : real-time priotity =0: normal scheduling >0: niced priority
Event	*wait_event	Event to be woken up or NULL

Table 11: Input Modifiers for schedule

Parameter Name	Parameter Type	Comment
Task	task_struct	

Table 12: Output Modifiers for schedule

Functional Specification:

3.4 Device Drivers

3.4.1 Motor driver

3.4.1.1 md_device_init

Parameter Name	Parameter Type	Comment
Device	device_id	

Table 13: Input Modifiers for md_device_init

Parameter Name	Parameter Type	Comment
ReturnCode	return_code	0 or error_code

Table 14: Output Modifiers for md_device_init

Functional Specification:

```
save Device.MotorState.mustep
set Device.MotorState.mustep = 0
call _md_set_phase(Device,0)
call md_microt_tick(Device,Phase) with Phase = (1...3)MAX_MUSTEPS...0
resttore Device.MotorState.mustep from saved Value
```

3.4.1.2 md_micro_tick

Parameter Name	Parameter Type	Comment
Device	device_id	
Phase	phase_spec	
Device.MotorState.mustep	boolean	
CurrentPhase	Static phase_spec	

Table 15: Input Modifiers for md_micro_tick

Parameter Name	Parameter Type	Comment
ReturnCode	return_code	0 or error_code

Table 16: Output Modifiers for md_micro_tick

Functional Specification:

```
If Phase  and CurrentPhase differ by more than MAX_MUSTEPS return EINVAL.
If Device.MotorState.mustep == FALSE and Phase % MAX_MUSTEPS != 0 return 0;
Call _md_set_phase(Device,Phase);
```

3.4.1.3 md_set_speed

Parameter Name	Parameter Type	Comment
Device	device_id	
Speed	speed_spec	
MotorDriverData.setspeed	boolean	

Table 17: Input Modifiers for md_set_speed

Parameter Name	Parameter Type	Comment
ReturnCode	return_code	0 or error_code

Table 18: Output Modifiers for md_set_speed

3.4.1.4 md_set_position

Parameter Name	Parameter Type	Comment
Device	device_id	
Position	Location_spec	
MotorDriverData.setposition	boolean	

Table 19: Input Modifiers for md_set_position

Parameter Name	Parameter Type	Comment
ReturnCode	return_code	0 or error_code

Table 20: Output Modifiers for md_set_position

3.4.1.5 md_set_targets

Parameter Name	Parameter Type	Comment
Device	device_id	
Position	position_spec	
MotorDriverData.settposition	boolean	
MotorDriverData.settposition	boolean	

Table 21: Input Modifiers for md_set_targets

Parameter Name	Parameter Type	Comment
ReturnCode	return_code	0 or error_code

Table 22: Output Modifiers for md_set_targets

3.4.1.6 _md_set_phase

Parameter Name	Parameter Type	Comment
Device	device_id	
Phase	phase_spec	

Table 23: Input Modifiers for md_micro_tick

Parameter Name	Parameter Type	Comment
ReturnCode	return_code	0 or error_code
CurrentPhase	phase_spec	

Table 24: Output Modifiers for md_micro_tick

Functional Specification:

If Phase and CurrentPhase differ by more than MAX_MUSTEPS return EINVAL.
If Device.MotorState.mustep == FALSE and Phase % MAX_MUSTEPS != 0 return 0;
Set phase of current Motor driver to Phase.

3.5 Application Space

3.5.1 Motor control

3.5.1.1 Variables

motorchanges	wait_event	
MotorState	motor_state[MAX_MOTORS]	

Table 25: Variables in Motor Control

3.5.1.2 Tasks

3.5.1.2.1 motor_task

Parameter Name	Parameter Type	Comment
Argc	1	Valid Motors
Argv[0]	"motors"	
Argv[1]	*Bitfield : MAX_MOTORS	
MotorState[].nextUpdate		
MotorState[].targets		

Table 26: Input Modifiers for motor_thread

Parameter Name	Parameter Type	Comment
ReturnCode	return_code	0 or error_code
NextUpdate	Static timespec	

Table 27: Output Modifiers for motor_thread

Functional Specification:

```
Continuosuly loop:
    Iterate over bits from Bitfield
        If Bit is not set → next bit
        adjust MotorState by calling device driver
        if MotorState.nextUpdate < NextUpdate → adjust NextUpdate
    schedule (NextUpdate,##,<0, motorchanged)
```

3.5.1.3 Functions

3.5.1.3.1 motor_set_targets

Parameter Name	Parameter Type	Comment
Device	device_id	
PositionData	position_spec	

Table 28: Input Modifiers for motor_set_target

Parameter Name	Parameter Type	Comment
ReturnCode	return_code	0 or error_code
MotorState.targets		

Table 29: Output Modifiers for motor_set_target

3.5.2 User Interface

4 Work Units

4.1 Staging

What	Who	When	Remarks

Table 30: Work Units and responsibilities

5 Indices

Index of Tables

Illustration Index

Alphabetical Index