Confidential

Specification Document

of YieldStar LPPS Control Module Mk4 Application Software

Prodrive B.V.

Document ID:	SPD6681170008R04.docx
Release date:	28-08-2018
Author(s):	Bas van de Ven
Status:	[Accepted]
Template ID:	SPDT120406

Document history

Rel.	Date	Changes
R01	2018-01-25	First version. Relevant changes wrt. SPD-6681-1500-08R11 marked in cyan.
R02	2018-01-30	Updated after internal review. Left cyan to indicate all changes w.r.t. to old specification to customer.
R03	2018-04-02	Updated after external review & integeration at Qioptiq -Changed Laser startup power from 60% to 35% -Updated screen shots -Updated timing delay for Cable & DC OK signals
R04	2018-08-28	Updated after customer requests & integration at customer site

Distribution list

Name	R01	R02	R03	R04							
Qioptiq											
Peter Jäger		x	x	x							
Prodrive											
Jordy Neis	x	x									
Bas van de Ven	x	x	x	x							
Yves Prevoo	x	x	x	x							
Tim Rebers			x	x							

Open issues

Issue	Section	Description	Responsible	Due Date

Contents

1. INTRODUCTION 8

1.1. BACKGROUND AND CONTEXT 8
1.2. DEFINITIONS AND ABBREVIATIONS 9
1.2.1. Definitions 9
1.2.2. Abbreviations 9

2. FEATURES 10

2.1. SOFTWARE OVERVIEW 10
2.2. FUNCTIONAL OVERVIEW 11

3. YLCM APPLICATION 15

3.1. DEFINITIONS 15
3.1.1. Sensor inputs 15
3.1.2. Actuator outputs 15
3.1.3. Fatal errors 15
3.1.4. Application state machine 16
3.1.5. Watchdog 16
3.1.6. Laser UART interface 17
3.2. COMMAND INTERFACE 17
3.2.1. Data types 17
3.2.2. Functions 18
3.2.2.1. GetFirmwareVersion 19
3.2.2.2. GetHardwareVersion 19
3.2.2.3. GetState 19
3.2.2.4. Reboot 20
3.2.2.5. ResetWatchdog 20
3.2.2.6. EnableMoves 20
3.2.2.7. DisableMoves 20
3.2.2.8. GetPositions 20
3.2.2.9. SetPositions 21
3.2.2.10. GetFilterTable 21
3.2.2.11. SetFilterTable 21
3.2.2.12. GetPumpLaserPower 22
3.2.2.13. SetPumpLaserPower 22
3.2.2.14. GetLampState 22
3.2.2.15. SetLampOn 22
3.2.2.16. SetLampOff 23
3.2.2.17. GetTotalLampOnTime 24
3.2.2.18. SetTotalLampOnTime 24
3.2.2.19. GetTotalPumpLaserOnTime 24
3.2.2.20. SetTotalPumpLaserOnTime 24
3.2.2.21. GetTelemetry 24
3.2.2.22. GetUptime 25
3.2.2.23. ReadTraceBuffer 25
3.2.2.24. SetPilotLaser 25
3.2.2.25. ReadNonVolatileMemory 25
3.2.2.26. WriteNonVolatileMemory 25
3.2.2.27. GetExtCableStatus 26
3.2.2.28. GetExtPsuStatus 26
3.2.2.29. GetQcmIdTime 26
3.2.2.30. GetLaserSerialNo 26
3.2.2.31. GetLaserCritErrorCode 27
3.2.2.32. GetLaserCritErrorCount 27
3.2.2.33. ResetLaserCritErrorCode 27
3.2.2.34. GetLaserSwVersion 27
3.2.2.35. GetLaserStatus 28
3.2.2.36. GetLaserOnTime 28
3.3. FIRMWARE UPLOAD 28

4. YLCM MK4 TEST APPLICATION 30

4.1. FUNCTIONAL REQUIREMENTS 30
4.2. NON-FUNCTIONAL REQUIREMENTS..30

5

List of Figures

Figure 1-1: LPPS Project scope 8
Figure 2-1: Overview functional blocks LPPS 10
Figure 2-2: Schematic view of the YLCM Mk4 and its connections 11
Figure 3-1: Application state machine 16
Figure 3-2: NSD flow of the SetLampOn sequence 23
Figure 3-3: Firmware upload (version numbers differ) mechanism 29
Figure 3-4: firmware upload (version numbers match) sequence 29
Figure 4-1: Example screenshot "Command functions" 31
Figure 4-2: Example screen "Software update" 32
Figure 4-3: Example screen "Trace buffer interface" 33
Figure 4-4: Example screen "Telemetry interface" 34

List of Tables

Table 2-1: Available YLCM sensors 12
Table 2-2: Available YLCM actuators 13
Table 3-1: Asynchronous fatal controller error descriptions 15
Table 3-2: Asynchronous fatal motor error descriptions 16
Table 3-3: Application state machine state descriptions 16
Table 3-4: Application state machine transition descriptions 16
Table 3-5: UART configuration 17
Table 3-6: functional command interface requirements 17
Table 3-7: Basic data types 17
Table 3-8: Derived data types 17
Table 3-9: Return value enumeration 18
Table 3-10: Message frame definition 18
Table 3-11: Overview of Payload command ID's and associated properties 18
Table 3-12: Payload parameters for GetFirmwareVersion 19
Table 3-13: Payload parameters for GetHardwareVersion 19
Table 3-14: Payload parameters for GetState 20
Table 3-15: Payload parameters for Reboot 20
Table 3-16: Payload parameters for ResetWatchdog 20
Table 3-17: Payload parameters for EnableMoves 20
Table 3-18: Payload parameters for DisableMoves 20
Table 3-19: Payload parameters for GetPositions 21
Table 3-20: Payload parameters for SetPositions 21
Table 3-21: Payload parameters for GetFilterTable 21
Table 3-22: Payload parameters for SetFilterTable 21
Table 3-23: Payload parameters for GetPumpLaserPower 22
Table 3-24: Payload parameters for SetPumpLaserPower 22
Table 3-25: Payload parameters for GetLampState 22
Table 3-26: Payload parameters for SetLampOn 22
Table 3-27: Payload parameters for SetLampOff 23
Table 3-28: Payload parameters for GetTotalLampOnTime 24
Table 3-29: Payload parameters for SetTotalLampOnTime 24
Table 3-30: Payload parameters for GetTotalPumpLaserOnTime 24
Table 3-31: Payload parameters for SetTotalPumpLaserOnTime 24
Table 3-32: Payload parameters for GetTelemetry 24
Table 3-33: Payload parameters for GetUptime 25
Table 3-34: Payload parameters for ReadTraceBuffer 25
Table 3-35: Payload parameters for SetPilotLaser 25
Table 3-36: Payload parameters for ReadNonVolatileMemory 25
Table 3-37: Payload parameters for WriteNonVolatileMemory 26
Table 3-38: Payload parameters for GetExtCableStatus 26
Table 3-39: Payload parameters for GetExtPsuStatus 26

Table 3-40: Payload parameters for GetQcmIdTime 26
Table 3-41: Payload parameters for GetLaserSerialNo 26
Table 3-42: Payload parameters for GetLaserCritErrorCode 27
Table 3-43: Payload parameters for GetLaserCritErrorCount 27
Table 3-44: Payload parameters for ResetLaserCritErrorCode 27
Table 3-45: Payload parameters for GetLaserSwVersion 27
Table 3-46: Payload parameters for GetLaserStatus 28
Table 3-47: Payload parameters for GetLaserOnTime 28

References

Mandatory

[1.1] SPD YieldStar LPPS Control Module Mk4
Author: Yves Prevoo
6681-1700-1102, 2018-01-22
SPD6681170011R02.pdf
http://orionfs:60001/getdoc?pn=6681170011

Referenced

[2.1] EPS LPPS TCP/IP protocol
Author: Jean Paul Mikkers
D000182374-03-EPS-001, 2013-01-03
D000182374-03-EPS-001.doc

[2.2] Dynamic Host Configuration Protocol
Author: R. Droms
http://tools.ietf.org/html/rfc2131, http://tools.ietf.org/html/rfc2132

[2.3] xLM-Series interface specification
Author: IPG photonics
G22-29629, 2017-12-27
G22-29629 rev2.pdf

[2.4] Datasheet: AT21CS11 1-Wire EEPROM
Author: Microchip (atmel)
DS20005857A, 2017-10-xx
DS20005857A.pdf

1. Introduction

This specification document describes the hardware for the ASML YieldStar Mk4 Source Module, this source module is a Laser Pumped Plasma Source (LPPS) developed by Qioptiq Photonics for Innovation. The module specification of the YLCM is available in the SPD YieldStar LPPS Control Module, reference [1.1].

The ASML YieldStar Mk4 Source Module (YCLM Mk4) is referenced in this document as ‘YLCM’. The corresponding YCLM Mk4 Backplane is referenced in this document as YCLM Backplane or YCLM-BP.

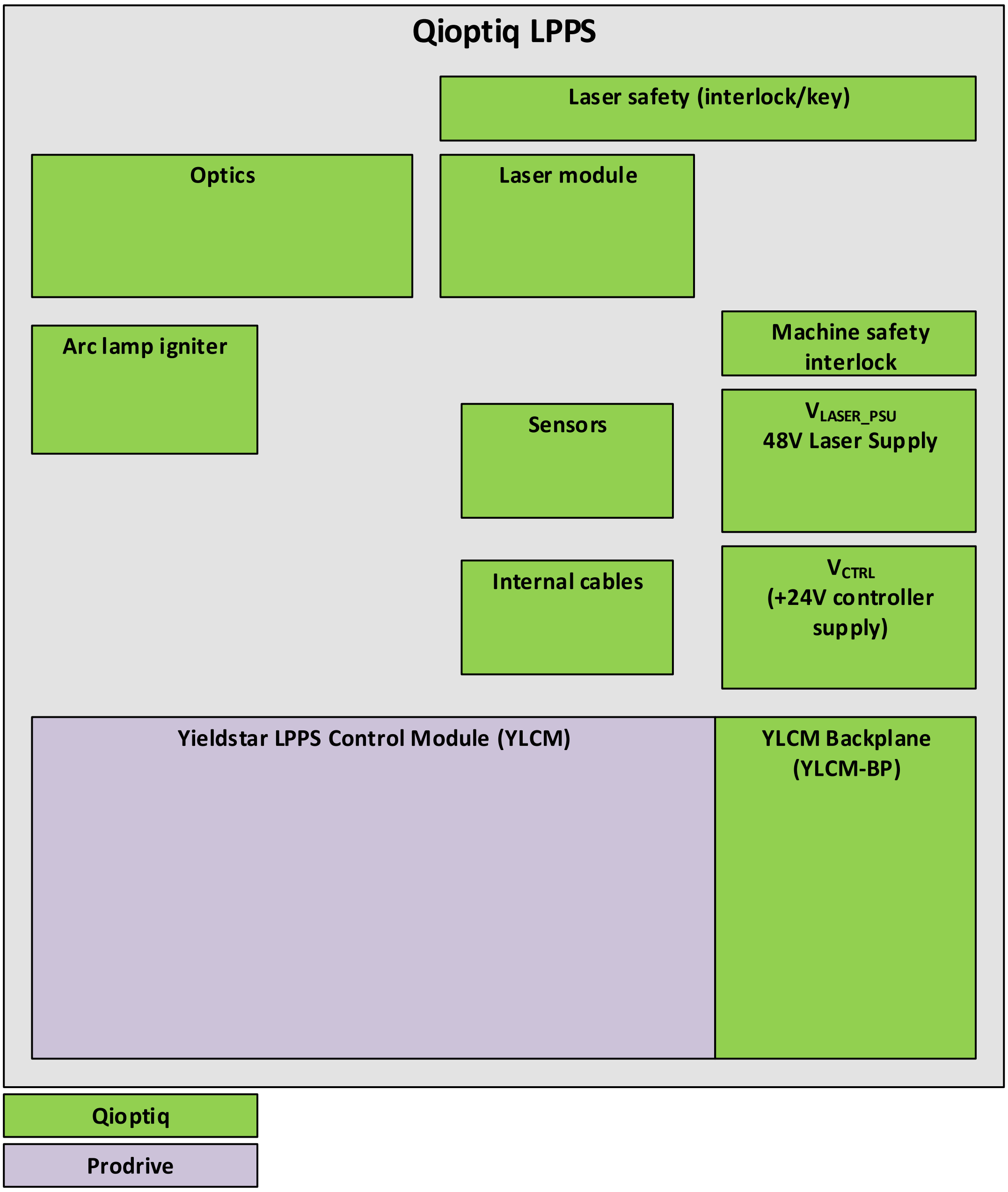


Figure 1-1: LPPS Project scope

1.1. Background and context

In the LPPS module light is generated by focussing a high power infrared laser beam into a silica bulb containing pressurized Xenon gas. Near this focal point hot plasma is generated radiating a high brightness broadband continuous spectrum of light. This light is then guided through selectable filters to output a specific spectral band of light.

1.2. Definitions and Abbreviations

1.2.1. Definitions

Marked text: Text marked in this color needs to be changed or completed.

Marked text: Text marked in this color has changed compared to the previous release.

Marked text: Text marked in this color is indicative and needs verification by measurements.

‘*a*’: Numeric binary notation (*a* can be multiple 0s or 1s). E.g. ‘010’ is a 3-bit value representing the binary number two. This kind of notation implies a specific bit length.

‘*aa.aaaa*’: Numeric binary notation with ‘.’ separations for clear reading of long binary numbers.

0x*a*: Numeric hexadecimal notation (*a* can be a digit 0 through 9, A through F). E.g. 0x1A is hexadecimal number twenty-six. This kind of notation does not directly imply a bit length.

0x*aa.aaaa* Numeric hexadecimal notation with ‘.’ separations for clear reading of long hexadecimal numbers.

ad: Numeric (explicit) decimal notation. This kind of notation does not directly imply a bit length.

X[b:a] Vector notation for vector X with bit range b downto a (little endian notation).

5

1.2.2. Abbreviations

BLDC	Brushless DC Motor
BSP	Board Support Package
CSV	Comma Separated Value
DC	Direct Current
DHCP	Dynamic Host Configuration Protocol
ENC	Encoder
FIFO	First-in-first-out
I/O	Input / Output
IR	Infrared
LED	Laser Emitting Diode
LPPS	Laser Pumped Plasma Source
LSB	Least significant Byte
MSB	Most significant Byte
NA	Not Applicable
NaN	Not a Number
NBF	Narrow Band Filter
NC	Not connected
NDF	Neutral Density Filter
NSD	Nassi–Shneiderman diagram
NVRAM	Non-volatile random access memory
OCP	Over current protection
OUI	Organizational Unique Identifier
OVP	Over voltage protection
PN	Product Number
QCM	Quick Change Module
RPC	Remote Procedure Call
SPD	Specification Document
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver Transmitter
UVP	Under voltage protection
WBF	Wide Band Filter
XML	eXtensible Markup Language
YLCM	YieldStar LPPS Control Module

2. Features

2.1. Software overview

The entire LPPS platform can be divided into several parts. These parts either run on an arbitrary client, on the application processor or on the Motion Controller. This division is shown in Figure 2.1.

Figure 2-1: Overview functional blocks LPPS

In short each block has the following functionalities/features:

YLCM TCP Application

- Handling of TCP/IP commands and translation to Motion controller calls
- Stating and control of the YLCM as a whole.

YLCM Test Application

- Windows based GUI access to all of the YLCM's functionality

Prodrive Motion Tooling

- Stand-alone application for uploading firmware to YLCM

Prodrive Motion API

- Multi-process API for handling RPC calls to the motion controller over TCP/IP

BSP – Ethernet / NVRAM

- Board support packages for accessing Ethernet & NVRAM

Prodrive Motion Controller

- Real-time interrupt based

- Handling of motor behavior (stating, commutation, position/current loop, etc)
- 3rd order motion profile trajectory generation
- Sampling ADC's, control Digital IO
- Cyclic signal acquisition (in combination with software scope in Prodrive Motion Tooling)

This document describes the interface functions of the LPPS application and the resulting process behavior.

2.2. Functional overview

A schematic representation along with the relevant input and output signals of the YLCM is given in Figure 2-2.

- IR Pump laser
  - Source of the laser beam
  - Contains low power "pilot laser" for testing/calibration purposes
  - RS232 UART interface
- Collimator
  - Narrows the beam of particles
- Quick Change Module
  - Interchangeable module containing pressurized xenon bulb
  - Generates continuous high brightness broadband plasma from IR laser
- Interlock status
  - Interlock output INT_STATUS (see [1.1]) reflects the state of the incoming INT_STATUS_IN signal
- Status LEDs

Figure 2-2: Schematic view of the YLCM Mk4 and its connections

The relevant sensor/actuator names based on [1.1] are described in Table 2-1 and Table 2-2.

Table 2-1: Available YLCM sensors

ID		Name	Description	Type	Unit
Group	Offset				
0x0000	0x0000	`IR_LASER_BACK_REFLECT`	Reflects the state of laser back reflection (reflection=1/no reflection=0) `IR_LASER_STATUS_CODE` bit 0	bool	Boolean [0 or 1]
	0x0001	`INT_QCM_PRESENT_STAT`	Reflects the state of the QCM (absent=1/present=0)	bool	Boolean [0 or 1]
	0x0002	`INT_COLLIMATOR_STAT`	Reflects the state of the Collimator (not in place=1/in place=0)	bool	Boolean [0 or 1]
	0x0003	`INT_DOOR_STAT`	Reflects the state of the door (open=1/closed=0)	bool	Boolean [0 or 1]
	0x0004	`reserved`			
	0x0005	`IR_LASER_DISABLE_STAT`	Reflects the state of the laser safety key (off=1/on=0)	bool	Boolean [0 or 1]
	0x0006	`INT_STATUS_IN`	Reflects the state of the incoming Interlock signal (open=1/closed=0)	bool	Boolean [0 or 1]
	0x0007	`VP_HKS_CMP`	Reflects the state of the V_{HKS} output (OCP circuit) comparator (error=1/ok=0)	bool	Boolean [0 or 1]
	0x0008	`VP_5V_CMP`	Reflects the state of the V_{5V_S} output (OCP circuit) comparator (error=1/ok=0)	bool	Boolean [0 or 1]
	0x0009	`VP_12V_CMP`	Reflects the state of the V_{12V_S} output (OCP circuit) comparator (error=1/ok=0)	bool	Boolean [0 or 1]
	0x000A	`IGN_STAT_PULSE`	Reflects the state of the igniter pulse feedback (active=0/inactive=1)	bool	Boolean [0 or 1]
	0x000B	`INT_SHUTTER_IN1`	Reflects the state of shutter 1 (ok=1/nok=0)	bool	Boolean [0 or 1]
	0x000C	`INT_SHUTTER_IN2`	Reflects the state of shutter 2 (ok=1/nok=0)	bool	Boolean [0 or 1]
	0x000D	`IR_LASER_READY`	Reflects the state of the laser status. `IR_LASER_STATUS_CODE` == 0	bool	Boolean [0 or 1]
	0x000E	`CONTROLLER_ERROR_CODE`	Error code currently active for controller. See Table 3-2 for bit mask.	uint32_t	Bitmask
	0x000F	`CONTROLLER_ERROR_ROOT_CODE`	First error code that occurred for controller. See Table 3-2 for bit mask. Cleared after 'DisableMoves'.	uint32_t	Bitmask
	0x0010	`IR_LASER_STATUS_CODE`	Status bit mask read via laser UART using "STA" command (also see [2.3])	uint32_t	Bitmask
	0x0011	`Reserved`			
	0x0012	`PSU1_CABLE_OK`	Reflects the state of PSU 1 cable connection (ok=1/nok=0) Latched at boot-up by toggling `CABLE_CHECK` and waiting 500 ms	bool	Boolean [0 or 1]
	0x0013	`PSU2_CABLE_OK`	Reflects the state of PSU 2 cable connection (ok=1/nok=0) Latched at boot-up by toggling `CABLE_CHECK` and waiting 500 ms	bool	Boolean [0 or 1]
	0x0014	`PSU1_DC_OK`	Reflects the state of the DC OK 1 input (ok=1/nok=0) Input should follow toggled inverted `IR_LASER_ENABLE` within 5 s Forced to 0 when `PSC1_CABLE_OK` is 0.	bool	Boolean [0 or 1]
	0x0015	`PSU2_DC_OK`	Reflects the state of the DC OK 2 input (ok=1/nok=0) Input should follow toggled inverted `IR_LASER_ENABLE` within 5 s Forced to 0 when `PSC2_CABLE_OK` is 0.	bool	Boolean [0 or 1]
	0x0016	`PSU1_TEMP_OK`	Reflects the state of the temperature 1 input (ok=1/nok=0) Input should follow toggled inverted `IR_LASER_ENABLE` within 5 s Forced to 0 when `PSC1_CABLE_OK` is 0.	bool	Boolean [0 or 1]
	0x0017	`PSU2_TEMP_OK`	Reflects the state of the temperature 2 input (ok=1/nok=0) Input should follow toggled inverted `IR_LASER_ENABLE` within 5 s Forced to 0 when `PSC2_CABLE_OK` is 0.	bool	Boolean [0 or 1]
	0x0018	`IR_LASER_FIBER`	Reflects the state of the fiber (broken=1/ok=0) `IR_LASER_STATUS_CODE`, bit 30	bool	Boolean [0 or 1]

0x0100	0x0000	TMP_COLLIMATOR	Analog temperature measurement Collimator	float	°C
	0x0001	TMP_QCM_BLOCK	Analog temperature measurement QCM Block	float	°C
	0x0002	TMP_BOARD_TOP	Analog temperature measurement YLCM board top	float	°C
	0x0003	TMP_BOARD_BOTTOM	Analog temperature measurement YLCM board bottom	float	°C
	0x0004	TMP_IR_LASER	Analog temperature measurement laser Read via laser UART using “RCT” command	float	°C
0x0200	0x0000	VOLTAGE_P24V_CONTROL	External 24V Control supply A value above 23.2V will enable “24V CTRL PSU” LED, see [1.1]	float	Volt
	0x0001	reserved			
	0x0002	VOLTAGE_P12V	Internal +12V supply	float	Volt
	0x0003	VOLTAGE_P13V	Internal +13V supply	float	Volt
	0x0004	VOLTAGE_N13V	Internal -13V supply	float	Volt
	0x0005	VOLTAGE_P5V	Internal +5V supply	float	Volt
	0x0006	VOLTAGE_P3V3	Internal +3.3V supply	float	Volt
	0x0007	VOLTAGE_P1V8	Internal +1.8V supply	float	Volt
	0x0008	VOLTAGE_P1V2	Internal +1.2V supply	float	Volt
	0x0009	reserved			
	0x000A	reserved			
	0x000B	OPT_LAMP_ON_DETECT	Lamp On feedback sensor	float	Volt
	0x000C	OPT_IR_DETECT	IR laser feedback sensor	float	Volt
	0x000D	VOLTAGE_P48V_LASER	IR laser supply feedback sensor A value above 45V will enable “48V LASER PSU” LED, see [1.1]	float	Volt
0x0300	0x0000	IR_LASER_POWER_OUT	IR laser power feedback Read via laser UART using “ROP” command	float	Watt
	0x0001	IR_LASER_PUMP_CURRENT	IR laser current feedback Read via laser UART using “RCS” command	float	Fraction
0x0800	0x0000	LAMP_TOTAL_ON_TIME	Persisted total on time of the lamp	time	[s,us]
	0x0001	IR_LASER_TOTAL_ON_TIME	Persisted total on time of the Infra-red laser	time	[s,us]
	0x0002	QCM_TOTAL_ON_TIME	Persisted total on time of QCM module	time	[s,us]

Table 2-2: Available YLCM actuators

Group	Offset	Name	Description	Type	Unit
0x0900	0x0000	IR_LASER_CONTROL	Controls analog power setpoint for the IR laser. Internally, this fraction is multiplied by *IrLaserControlGain*, if *IrLasterControlGain* is not equal to zero (also see 4.1). Written via laser UART using “SDC” command.	float	Fraction
	0x0001	IR_LASER_ON_OFF	Control Infra-red laser (on=1/off=0) Written via laser UART using “EMON”/”EMOFF” commands	bool	Boolean [0 or 1]
	0x0002	PILOT_LASER_ON_OFF	Controls lower power pilot laser for testing/debugging purposes (on=1/off=0) Written via laser UART using “ABN”/”ABF” commands	bool	Boolean [0 or 1]
	0x0003	LAMP_START	Enable igniter (on=1/off=0), used during startup of the lamp	bool	Boolean [0 or 1]

	0x0004	IR_LASER_ENABLE	Laser supply control output (on=1/off=0)	bool	Boolean [0 or 1]
	0x0005	IR_LASER_RESET	Laser reset, Written via laser UART using “RERR” command	bool	Boolean [0 or 1]
	0x0006	CABLE_CHECK	Cable check control output. Toggled at startup to latch PSU1_CABLE_OK and PSU2_CABLE_OK.	bool	Boolean [0 or 1]

3. YLCM Application

3.1. Definitions

The naming conventions for the YLCM application throughout the document are described in the following chapters.

3.1.1. Sensor inputs

The sensors for the application are connected to the inputs of the YLCM (for a complete list see Table 2-1).

3.1.2. Actuator outputs

The actuators for the application are connected to the outputs of the YLCM (for a complete list see Table 2-2).

3.1.3. Fatal errors

The YLCM motion controller continuously monitors the state of the controller as a whole and each individual motor. Presence of such asynchronous errors will result in a direct transition to the “error” state of the application state machine (see 3.1.4). In this state the following actions are executed asynchronously:

- Lamp & laser are disabled, by clearing LAMP_START, IR_LASER_ON_OFF, IR_LASER_ENABLE and setting IR_LASER_CONTROL setpoint to 0.

Certain errors can invalidate the encoder position which means that alignment & homing of the motor must be executed during the next *EnableMove* command (see 3.2.2.6).

Table 3-1: Asynchronous fatal controller error descriptions

Name	Error code	Description
ControlSupplyUvp	0x00000001	24V control supply under voltage protection
ControlSupplyOvp	0x00000002	24V control supply over voltage protection
~~LaserSupplyUvp~~	~~0x00000004~~	~~48V laser supply under voltage protection~~
~~LaserSupplyOvp~~	~~0x00000008~~	~~48V laser supply over voltage protection~~
InternalVoltage	0x00000010	Internal voltage supply error (OVP or UVP)
BoardOtp	0x00000020	Over temperature trip on YLCM on-board temperature sensor
LaserBackReflection	0x00000040	IR_LASER_BACK_REFLECT is high
IrPowerOpticalPath	0x00000080	OPT_IR_DETECT above *IrPowerOpticalThreshold* level, while the lamp has been enabled, configurable via Test application (see 4.1).
LampOpticalPath	0x00000100	OPT_LAMP_ON_DETECT below *LampOpticalThreshold* level, while the lamp has been enabled, configurable via Test application (see 4.1).
IrPowerOpticalDisconnect	0x00000200	OPT_IR_DETECT below *IrPowerOpticalDisconnect* level configurable via Test application (see 4.1). If *IrPowerOpticalDisconnect* is 0 this error is disabled
LampOpticalDisconnect	0x00000400	OPT_LAMP_ON_DETECT below *LampOpticalDisconnect* level, configurable via Test application (see 4.1). If *LampOpticalDisconnect* is 0 this error is disabled
TmpQcmBlockOverLimit	0x00000800	TMP_QCM_BLOCK has exceeded the *TmpQcmBlockThreshold* level configurable via Test application (see 4.1). If *TmpFocusBlockThreshold* is 0 this error is disabled
TmpCollimatorOverLimit	0x00001000	TMP_COLLIMATOR has exceeded the *TmpCollimatorThreshold* level configurable via Test application (see 4.1). If *TmpCollimatorThreshold* is 0 this error is disabled
InterlockOpen	0x00004000	interlock circuit interrupted (i.e. INT_STATUS_IN is high)
FirmwareWatchdog	0x00008000	Motion controller not running (controller interrupt loop absent)
CpuLoad	0x00010000	Controller interrupt cycle CPU load is too high
Psu1CableDisconnect	0x00100000	PSU1_CABLE_OK is low, when CABLE_CHECK active
Psu2CableDisconnect	0x00200000	PSU2_CABLE_OK is low, when CABLE_CHECK active
Psu1DcNok	0x00400000	PSU1_DC_OK doesn’t follow IR_LASER_ENABLE
Psu2DcNok	0x00800000	PSU2_DC_OK doesn’t follow IR_LASER_ENABLE
Psu1TempNok	0x01000000	PSU1_TEMP_OK doesn’t follow IR_LASER_ENABLE

Psu2TempNok	0x02000000	PSU2_TEMP_OK doesn't follow IR_LASER_ENABLE
LaserColFiber	0x04000000	STA bit 30 optional IR_LASER_FIBER
LaserUartComm	0x08000000	Laser communication failed (response timeout)

Table 3-2: Asynchronous fatal motor error descriptions

Name	Error code	Invalidate encoder	Description
-			

3.1.4. Application state machine

The application state machine of the YLCM application is illustrated in Figure 3-1.

- When a state has multiple outputs to other states the priority is denoted with A, B, etc. where A has the highest priority
- Commands (see 3.2) that influence the state machine are denoted with "Cmd"
- Errors are defined in Table 3-1 & Table 3-2

Figure 3-1: Application state machine

Description of each state is given in Table 3-3.

Table 3-3: Application state machine state descriptions

Name	Module Status LED	Description
Initial	Off	Initial state after application software start
Startup	Green Blinking	Startup sequence
ReadyToMove	Green	Lamp control possible
Moving	Green	Busy with processing a queued command (e.g. SetLampOn)
Error	Amber (Red+Green)	A fatal error occurred (e.g. interlock) Lamp are disabled

Table 3-4: Application state machine transition descriptions

Name	Section	Description
Init	-	Initialization of YLCM completed (boot up)
EnableMoves	3.2.2.6	Executes startup sequence.
DisableMoves	3.2.2.7	Shuts down the lamp and laser
SetLampOn	3.2.2.15	
Errors detected	3.1.3	A fatal error occurred

3.1.5. Watchdog

The application software is equipped with a Watchdog timer that is used for checking whether there is at least still one valid connection with the user application. The watchdog:

- Is reset when a valid command (see Table 3-11) is received
- Is frozen while a command is busy, or when not in the "ReadyToMove" state
- Expires after 60 seconds

3.1.6. Laser UART interface

The UART which is connected to the laser unit is configured as following shown in Table 3-5.

Table 3-5: UART configuration

Parameter	Value
Baud rate	57600 baud
Data format	8N1: 8 data bits, no parity, 1 stop bit, no flow control

IR_LASER_STATUS_CODE is updated internally every 5 seconds by polling laser UART with “STA” command (also see [2.3]). This is used for autonomous error detection.

3.2. Command Interface

The interface provides a method of communication with the YLCM over Ethernet. This interface must comply with the following requirements as given in Table 3-6.

Table 3-6: functional command interface requirements

Requirement	Description
1	The YLCM supports IPv4
2	The YLCM’s MAC address shall have fixed 24-bit OUI: 00-0F-11
3	The YLCM acquires its IP address using DHCP (RFC 2131, also see [2.2])
4	The YLCM supplies “LPPSMK4” as hostname to the DHCP server, using ‘hostname’ option
5	The YLCM responds to ping requests, once an IP address is allocated
6	The YLCM listens for TCP/IP connections on port: 20020
7	The YLCM can handle and maintain at least 4 concurrent TCP/IP connections
8	The YLCM can handle fragmented transmission of commands specified in 3.2.2
9	The YLCM disables “Nagle coalescing algorithm” on its TCP/IP sockets connected to the user application (i.e. TCP_NODELAY is set to ‘1’)
10	The YLCM handles incoming commands in serialized fashion per TCP/IP connection (next command handled after response is returned).

3.2.1. Data types

Table 3-7 describes the basic data types used in the command interface.

Table 3-7: Basic data types

Name	# bytes	Description
bool	1	Boolean (0 = false, 1 = true)
int8_t	1	Signed 8-bit
int16_t	2	Signed 16-bit
int32_t	4	Signed 32-bit
uint8_t	1	Unsigned 8-bit
uint16_t	2	Unsigned 16-bit
uint32_t	4	Unsigned 32-bit
uint64_t	8	Unsigned 64-bit
float	4	IEEE-754 Single precision floating point (binary32)
double	8	IEEE-754 Double precision floating point (binary64)

Table 3-8 describes the data types that are derived from the basic types as given in Table 3-7 possibly with variable length.

Table 3-8: Derived data types

Name	(Sub)Type	Description
string	uint16_t	Size “N” of string plus 0 terminator [byte]
	int8_t[N]	UTF-8 zero terminated string
time	uint32_t	Time [s]
	uint32_t	Fractional time [us]
filter	uint8_t	FilterWheel index
	uint8_t	FilterSlot index
	uint8_t	Filter Type. Safe/Unsafe slot marking is not used by YLCM. 0x0: Unknown (Unsafe) 0x1: Open (Unsafe) 0x2: Closed (Safe) 0x3: Bandpass (Safe) 0x4: Attenuator (Unsafe)

		0x5: Focus calibration (Safe) 0x6 – 0xFF: Reserved (Safe)
	float	Nominal wave length [nm] (typically only used for band pass filters)
	float	Measured wave length [nm]
	float	Bandwidth [nm]
	float	Attentuator value (typically, only used for attenuator filters)
	uint8_t[128]	User meta data
traceline	time	Trace line timestamp
	string	Trace line string

Table 3-9: Return value enumeration

Value	Name	Description
0x0	ResultOk	No error detected
0x1	ResultUnknownCmd	Received command ID doesn't match any of the existing numbers (see Table 3-11)
0x2	ResultInvalidFormat	Number of payload bytes doesn't match command, Argument value out of range
0x3	ResultInvalidState	Command not allowed in current state
0x4	ResultTimeout	Command timeout expiration
0x5	ResultMoveFailure	Error occurred in one of the move sequences
0x6	ResultRebootBusy	After initiating a reboot sequence all function calls that are received, before the actual reboot will return this error instead of executing the function.
0x7	ResultNoIgniter	A required device is disconnected
0x8	ResultNoLamp	Ignition succeeded, but no lamp detected
0x9	ResultIgnitionFailStart	The igniter is not functioning correctly
0xA	ResultIgnitionFailCharge	The igniter is not functioning correctly, see Figure 3-2
0xB	ResultIgnitionIrSensor	The igniter is not functioning correctly, see Figure 3-2
0xC	ResultLaserNotReady	The laser is not ready for lamp on sequence, see Figure 3-2

3.2.2. Functions

Communication between user application and YLCM application, i.e. commands (in) and replies (out), are based on the following raw TCP/IP message frame definition as defined in Table 3-10. All basic data types are received/transferred in network order (big endian format; MSB first). For response commands it holds that any additional parameters after the failure return code may be omitted, as long as the header's Length field is correct.

Table 3-10: Message frame definition

Section	Byte	Name	Data type	Description
Header	0	Start preamble	uint16_t	Synchronization word to denote start of a valid data frame. Contains fixed value 0xAAAA.
	1			
	2	Protocol ID	uint8_t	Version identification for the message protocol used. Used to match interface definition between user and LPPS application. Contains fixed value 0x1.
	3	Length	uint16_t	Size "N" bytes of the following payload.
	4			
Payload	0	Command ID	uint8_t	First payload byte always contains the Command ID. See Table 3-11 for an overview of all possible commands and their values.
	1	Parameter Byte 1	Var.	Second payload byte. Depending on the command and in/out direction this may contains parameters for the corresponding command and/or response
	…			
	N-1	Parameter Byte N-1	Var.	Last payload byte for the respective command.

Table 3-11 specifies all commands available to the YLCM application.

- Concurrency denotes whether multiple TCP/IP connections can call this at the same time. The YLCM application will guarantee data integrity when multiple commands for a particular function are called. Commands that involve motor movement must be executed sequentially, and are thus put in a FIFO queue before getting executed.
- Precondition state determines whether the command is allowed in the particular state. As "Startup" & "Moving" denote busy/blocked states for queued command, any such command will remain in the queue until the current command has completed.

Table 3-11: Overview of Payload command ID's and associated properties

ID	Name	Section	Concurrency	Precondition state
0x01	GetFirmwareVersion	3.2.2.1	Yes	Any state
0x02	GetHardwareVersion	3.2.2.2	Yes	Any state

0x03	GetState	3.2.2.3	Yes	Any state
0x04	Reboot	3.2.2.4	Queued	Any state
0x05	ResetWatchdog	3.2.2.5	Yes	Any state
0x06	EnableMoves	3.2.2.6	Queued	Initial
0x07	DisableMoves	3.2.2.7	Queued	ReadyToMove, Startup, Moving, Error
0x08	GetPositions	3.2.2.8	Yes	Any state
0x09	SetPositions	3.2.2.9	Queued	ReadyToMove, Startup, Moving
0x0A	GetFilterTable	3.2.2.10	Yes	Any state
0x0B	SetFilterTable	3.2.2.11	Queued	Initial
0x0C	GetPumpLaserPower	3.2.2.12	Yes	Any state
0x0D	SetPumpLaserPower	3.2.2.13	Queued	ReadyToMove, Startup, Moving
0x0E	GetLampState	3.2.2.14	Yes	Any state
0x0F	SetLampOn	3.2.2.15	Queued	ReadyToMove, Startup, Moving
0x10	SetLampOff	3.2.2.16	Queued	Any state
0x11	GetTotalLampOnTime	3.2.2.17	Yes	Any state
0x12	SetTotalLampOnTime	3.2.2.18	Queued	Any state
0x13	GetTotalPumpLaserOnTime	3.2.2.19	Yes	Any state
0x14	SetTotalPumpLaserOnTime	3.2.2.20	Queued	Any state
0x15	GetTelemetry	3.2.2.21	Yes	Any state
0x16	GetUptime	3.2.2.22	Yes	Any state
0x17	ReadTraceBuffer	3.2.2.23	Yes	Any state
0x18	SetPilotLaser	3.2.2.24	Yes	Any state
0x19	ReadNonVolatileMemory	3.2.2.25	Yes	Any state
0x1A	WriteNonVolatileMemory	3.2.2.26	Queued	Initial
0x1B	GetExtCableStatus	3.2.2.27	Yes	Any state
0x1C	GetExtPsuStatus	3.2.2.28	Yes	Any state
0x1D	GetQcmIdTime	3.2.2.29	Yes	Any state
0x1E	GetLaserSerialNo	3.2.2.30	Yes	Any state
0x1F	GetLaserCritErrorCode	3.2.2.31	Yes	Any state
0x20	GetLaserCritErrorCount	3.2.2.32	Yes	Any state
0x21	GetLaserCritError	3.2.2.33	Yes	Any state
0x22	GetLaserSwVersion	3.2.2.34	Yes	Any state
0x23	GetLaserStatus	3.2.2.35	Yes	Any state
0x24	GetLaserOnTime	3.2.2.36	Yes	Any state

3.2.2.1. GetFirmwareVersion

GetFirmwareVersion retrieves firmware version that is currently running on the YLCM.

Table 3-12: Payload parameters for GetFirmwareVersion

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: no errors
	uint8_t	Major number	Release number (last 2 digits of the PN): Mk1: 00-19 Mk2: 20-49 Mk3 FUMO: 50-69 Mk3: 70-89 Mk4: 90-109
	uint8_t	Minor number	Subrelease number

3.2.2.2. GetHardwareVersion

GetHardwareVersion retrieves the hardware version for the YLCM.

Table 3-13: Payload parameters for GetHardwareVersion

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: no errors
	uint8_t	Major number	Release number (last 2 digits of the PN)
	uint8_t	Minor number	Backplane ID (see [1.1]).

3.2.2.3. GetState

GetState retrieves the current state of the YLCM application.

Table 3-14: Payload parameters for GetState

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: no errors
	uint8_t	State	See 3.1.4 for an overview of the application state machine. 0x0: Initial 0x1: Startup 0x2: ReadyToMove 0x3: Moving 0x4: Error

3.2.2.4. Reboot

Reboot enforces a soft reboot on the firmware running on the YLCM. This command will call *DisableMoves* internally to make sure all motors & lamp are disabled. After doing so, the YLCM will reboot with a delay of 1 second. This to ensure the response frame can be read out by the user application.

Table 3-15: Payload parameters for Reboot

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: no errors

3.2.2.5. ResetWatchdog

ResetWatchdog acts as a ping command and is required to be called (or any other command) to reset the application's watchdog timer, when being idle in "ReadyToMove" state. Behavior of the watchdog is given in section 3.1.5.

Table 3-16: Payload parameters for ResetWatchdog

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: no errors ResultRebootBusy: Reboot sequence in progress

3.2.2.6. EnableMoves

EnableMoves acts as a fallthrough.

Table 3-17: Payload parameters for EnableMoves

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: no errors ResultInvalidState: Cannot be called in this state ResultRebootBusy: Reboot sequence in progress

3.2.2.7. DisableMoves

DisableMoves turns off the lamp (if not off already). Furthermore, the laser is reset with "RERR" command to reset all errors (also see [2.3]).

Table 3-18: Payload parameters for DisableMoves

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: no errors ResultInvalidState: Cannot be called in this state ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress

3.2.2.8. GetPositions

GetPositions returns the currently selected filter slot index for all filter wheels.

Table 3-19: Payload parameters for GetPositions

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: no errors ResultRebootBusy: Reboot sequence in progress
	uint8_t	Filter0SlotIndex	Selected filter slot index value 0xFF (invalid) will always be returned.
	uint8_t	Filter1SlotIndex	
	uint8_t	Filter2SlotIndex	
	uint8_t	Filter3SlotIndex	

3.2.2.9. SetPositions

SetPositions is not supported, and returns ResultInvalidState.

Table 3-20: Payload parameters for SetPositions

Parameter	Type	Name	Description
In	uint8_t	Filter0SlotIndex	Selected filter slot index for each for filterwheel
	uint8_t	Filter1SlotIndex	
	uint8_t	Filter2SlotIndex	
	uint8_t	Filter3SlotIndex	
Out	uint8_t	Return value	ResultOk: no errors ResultInvalidState: Cannot be called in this state ResultRebootBusy: Reboot sequence in progress

3.2.2.10. GetFilterTable

GetFilterTable returns all the stored filter wavelength data as persisted in NVRAM. Data for all filter wheels will be returned.

Table 3-21: Payload parameters for GetFilterTable

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: no errors ResultRebootBusy: Reboot sequence in progress
	uint16_t	Size	Size “N” elements in the array
	filter[N]	Filter data	Array of structures containing data for each filter wheels (see Table 3-8). Data for all available filter indexes will be returned. *Note: If no value has been programmed yet, all filters will return unknown as filter type.*

3.2.2.11. SetFilterTable

SetFilterTable writes the given set of wave lengths as data in NVRAM. This user data is only stored for retrieval with *GetFilterTable*. Missing filter indices may be incomplete; any missing items in this list should be marked as FilterType “Unknown”.

The values of the “FilterType” are checked for validness. Other fields are treated as data and are only directly stored into NVRAM.

Table 3-22: Payload parameters for SetFilterTable

Parameter	Type	Name	Description
In	uint16_t	Size	Size “N” elements in the array
	filter[N]	Filter data	Array of structures containing data for each filter wheels (see Table 3-8).
Out	uint8_t	Return value	ResultOk: No errors ResultInvalidFormat: FilterWheelIndex, FilterIndex, FilterType ResultInvalidState: Cannot be called in this state ResultRebootBusy: Reboot sequence in progress

3.2.2.12. GetPumpLaserPower

GetPumpLaserPower retrieves the current laser power setpoint. This analog value will retrieved from IR_LASER_CONTROL. To retrieve measured feedback value from the IR_LASER_POWER_OUT sensor use *GetTelemetry* (see 3.2.2.21).

Table 3-23: Payload parameters for GetPumpLaserPower

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress
	float	Laser power	Measured laser power as fraction of nominal output power [fraction] by dividing the returned power in Watts / 700. When “ROP” command returns “low” / “off” value 0.0 will be returned.

3.2.2.13. SetPumpLaserPower

SetPumpLaserPower sets the current laser power setpoint. It will automatically be set to zero when a fatal error occurs (see 3.1.3). This function updates IR_LASER_CONTROL, which is the setpoint that is transferred over UART with “SDC” command (also see [2.3]). If the laser is off the setpoint is only cached in the YLCM and will become active during the startup sequence in the “SetLampOn” sequence.

Table 3-24: Payload parameters for SetPumpLaserPower

Parameter	Type	Name	Description
In	float	Laser power	Requested laser power relative of nominal output power [fraction]
Out	uint8_t	Return value	ResultOk: No errors ResultInvalidState: Cannot be called in this state ResultInvalidFormat: laser power value invalid or outside acceptable range ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress

3.2.2.14. GetLampState

GetLampState retrieves the current status of the Lamp feedback with OPT_LAMP_ON_DETECT sensor, and compares it to *LampOpticalThreshold* parameter.

Table 3-25: Payload parameters for GetLampState

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultRebootBusy: Reboot sequence in progress
	uint8_t	Status	0x0: Off 0x1: On

3.2.2.15. SetLampOn

The *SetLampOn* requests enabling of the lamp. The t_x & mode parameters can be persisted using the windows based test application (see 4).

Table 3-26: Payload parameters for SetLampOn

Parameter	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultInvalidState: Cannot execute this function in this state ResultRebootBusy: Reboot sequence in progress ResultTimeout: Lamp does not remain enabled after timeout expires ResultLaserNotReady: laser status indicates that laser is not ready for startup ResultNoIgniter: The ignition unit is not connected/detected ResultNoLamp: No lamp detected during start sequence ResultIgnitionFailCharge: No rising edge detected on ignitor ResultIgnitionIrSensor: IR_OPT_DETECT above IrPowerOpticalThreshold

The value of "mode" determines which sequence is executed during *SetLampOn* (if equal to 0 with igniter and otherwise without igniter). Both sequences are shown in Figure 3-2.

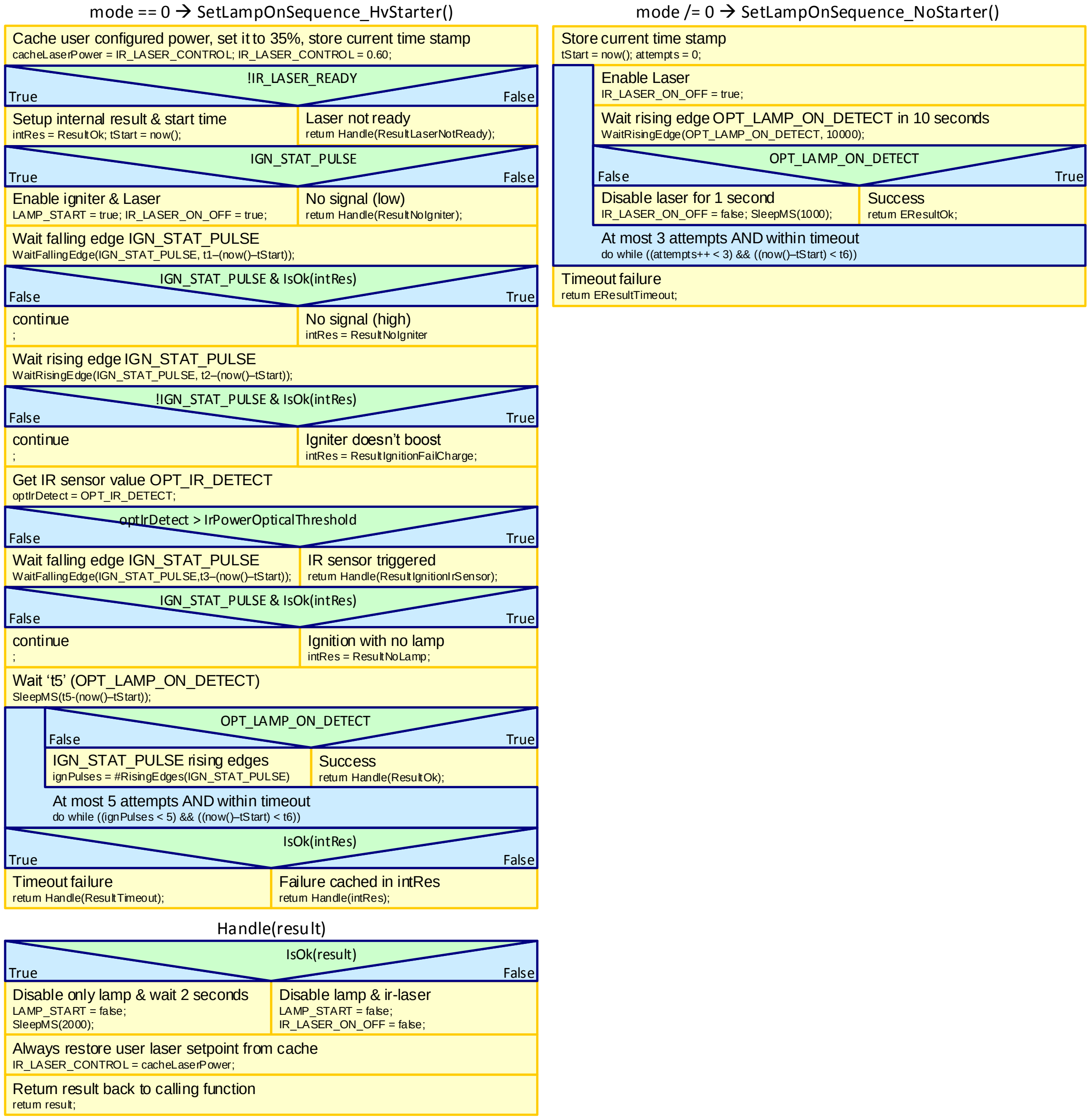

mode == 0 → SetLampOnSequence_HvStarter()
Cache user configured power, set it to 35%, store current time stamp
cacheLaserPower = IR_LASER_CONTROL; IR_LASER_CONTROL = 0.60;
!IR_LASER_READY
True
False
Setup internal result & start time
intRes = ResultOk; tStart = now();
Laser not ready
return Handle(ResultLaserNotReady);
IGN_STAT_PULSE
True
False
Enable igniter & Laser
LAMP_START = true; IR_LASER_ON_OFF = true;
No signal (low)
return Handle(ResultNoIgniter);
Wait falling edge IGN_STAT_PULSE
WaitFallingEdge(IGN_STAT_PULSE, t1–(now()–tStart));
IGN_STAT_PULSE & IsOk(intRes)
False
True
continue
;
No signal (high)
intRes = ResultNoIgniter
Wait rising edge IGN_STAT_PULSE
WaitRisingEdge(IGN_STAT_PULSE, t2–(now()–tStart));
!IGN_STAT_PULSE & IsOk(intRes)
False
True
continue
;
Igniter doesn't boost
intRes = ResultIgnitionFailCharge;
Get IR sensor value OPT_IR_DETECT
optIrDetect = OPT_IR_DETECT;
optIrDetect > IrPowerOpticalThreshold
False
True
Wait falling edge IGN_STAT_PULSE
WaitFallingEdge(IGN_STAT_PULSE,t3–(now()–tStart));
IR sensor triggered
return Handle(ResultIgnitionIrSensor);
IGN_STAT_PULSE & IsOk(intRes)
False
True
continue
;
Ignition with no lamp
intRes = ResultNoLamp;
Wait 't5' (OPT_LAMP_ON_DETECT)
SleepMS(t5-(now()–tStart));
OPT_LAMP_ON_DETECT
False
True
IGN_STAT_PULSE rising edges
ignPulses = #RisingEdges(IGN_STAT_PULSE)
Success
return Handle(ResultOk);
At most 5 attempts AND within timeout
do while ((ignPulses < 5) && ((now()–tStart) < t6))
IsOk(intRes)
True
False
Timeout failure
return Handle(ResultTimeout);
Failure cached in intRes
return Handle(intRes);
Handle(result)
IsOk(result)
True
False
Disable only lamp & wait 2 seconds
LAMP_START = false;
SleepMS(2000);
Disable lamp & ir-laser
LAMP_START = false;
IR_LASER_ON_OFF = false;
Always restore user laser setpoint from cache
IR_LASER_CONTROL = cacheLaserPower;
Return result back to calling function
return result;
mode /= 0 → SetLampOnSequence_NoStarter()
Store current time stamp
tStart = now(); attempts = 0;
Enable Laser
IR_LASER_ON_OFF = true;
Wait rising edge OPT_LAMP_ON_DETECT in 10 seconds
WaitRisingEdge(OPT_LAMP_ON_DETECT, 10000);
OPT_LAMP_ON_DETECT
False
True
Disable laser for 1 second
IR_LASER_ON_OFF = false; SleepMS(1000);
Success
return EResultOk;
At most 3 attempts AND within timeout
do while ((attempts++ < 3) && ((now()–tStart) < t6))
Timeout failure
return EResultTimeout;

Figure 3-2: NSD flow of the SetLampOn sequence

3.2.2.16. SetLampOff

SetLampOff disables the lamp & laser over UART with "EMOFF" command (also see [2.3]).

Table 3-27: Payload parameters for SetLampOff

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultTimeOut: Lamp sensor remains active after 3 seconds. ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress

3.2.2.17. GetTotalLampOnTime

GetTotalLampOnTime retrieves the time the lamp has been enabled from NVRAM. When the lamp is on this value will be maintained and persisted by the YLCM application every 1 minute (value may wrap-around).

Table 3-28: Payload parameters for GetTotalLampOnTime

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultRebootBusy: Reboot sequence in progress
	time	on time	Total time the lamp has been enabled

3.2.2.18. SetTotalLampOnTime

SetTotalLampOnTime sets a specified time as the current total on time in NVRAM.

Table 3-29: Payload parameters for SetTotalLampOnTime

Payload	Type	Name	Description
In	time	on time	Time lamp has been enabled
Out	uint8_t	Return value	ResultOk: No errors ResultRebootBusy: Reboot sequence in progress

3.2.2.19. GetTotalPumpLaserOnTime

GetTotalPumpLaserOnTime retrieves the time the IR laser has been enabled from NVRAM. When the pump laser is on this value will be maintained and persisted by the YLCM application every 1 minute (value may wrap-around).

Table 3-30: Payload parameters for GetTotalPumpLaserOnTime

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultRebootBusy: Reboot sequence in progress
	time	on time	Time IR laser has been enabled

3.2.2.20. SetTotalPumpLaserOnTime

SetTotalPumpLaserOnTime sets a specified time as the current total on time in NVRAM.

Table 3-31: Payload parameters for SetTotalPumpLaserOnTime

Payload	Type	Name	Description
In	time	on time	Time IR laser has been enabled
Out	uint8_t	Return value	ResultOk: No errors ResultRebootBusy: Reboot sequence in progress

3.2.2.21. GetTelemetry

GetTelemetry retrieves the requested data by index as defined in 2.2. If there is no UART communication possible with the Laser instead of failing this function call the following values for telemetry elements are returned:

• IR_LASER_STATUS_CODE	0x1 (Laser comm buffer overflow)
• IR_LASER_POWER_OUT	0
• IR_LASER_PUMP_CURRENT	0
• TMP_IR_LASER	0

Table 3-32: Payload parameters for GetTelemetry

Frame	Type	Name	Description
In	uint16_t	Size	Size “N” elements in the array
	uint16_t[N]	TelemetryID	See Table 2-1, Table 2-2 for valid ID numbers

Out	uint8_t	Return value	ResultOk: No errors ResultInvalidFormat: Unknown/invalid Telemetry specified ResultRebootBusy: Reboot sequence in progress
	uint8_t[N]	Data	Returned data parameters have variable width depending on the selected TelemetryID's

3.2.2.22. GetUptime

GetUptime retrieves the time the YLCM has been booted.

Table 3-33: Payload parameters for GetUptime

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultRebootBusy: Reboot sequence in progress
	time	uptime	Time elapsed since last boot.

3.2.2.23. ReadTraceBuffer

ReadTraceBuffer retrieves the internal logging buffer of the application.

Table 3-34: Payload parameters for ReadTraceBuffer

Payload	Type	Name	Description
In	uint16_t	Size max	Maximum allowed number of trace lines to return
Out	uint8_t	Return value	ResultOk: No errors ResultRebootBusy: Reboot sequence in progress
	uint32_t	Size remaining	Number of trace line entries remaining after returning this function
	uint32_t	Size lost	Number of trace lines lost due to overflow of internal circular buffer
	uint16_t	Size	Size "N" traceline elements in the array, will always be smaller or equal to "Size max" input parameter.
	trace line[N]	Trace	Trace timestamp and string, returned in order from old first to new last.

3.2.2.24. SetPilotLaser

SetPilotLaser enables the pilot laser beam. This can be used as a guidance to align the filter wheels for an operator. Since the pilot laser falls outside the error stating, it can be enabled in any state, and is controlled through UART with "ABN" / "ABF" commands (also see [2.3]).

Table 3-35: Payload parameters for SetPilotLaser

Payload	Type	Name	Description
In	bool	State	Enables/disables the Pilot laser beam
Out	uint8_t	Return value	ResultOk: No errors ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress

3.2.2.25. ReadNonVolatileMemory

ReadNonVolatileMemory retrieves the internal NVRAM data of the YLCM as a single binary.

Table 3-36: Payload parameters for ReadNonVolatileMemory

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultRebootBusy: Reboot sequence in progress
	uint16_t	Size	Size "N" elements in the array
	uint8_t[N]	Buffer	NVRAM buffer contents

3.2.2.26. WriteNonVolatileMemory

WriteNonVolatileMemory persists the internal NVRAM data of the YLCM as a single binary. The internally stored version information will ensure that writing an older versioned buffer is updated before it is persisted. However, it is not possible to store a newer versioned buffer in older firmware. Uploading an incompatible Lpps250 buffer with also result in a ResultFail.

Table 3-37: Payload parameters for WriteNonVolatileMemory

Payload	Type	Name	Description
In	uint16_t	Size	Size “N” elements in the array
	uint8_t[N]	Buffer	NVRAM buffer contents
Out	uint8_t	Return value	ResultOk: No errors ResultFail: Buffer is invalid/incompatible ResultInvalidState: Cannot execute this function in this state ResultRebootBusy: Reboot sequence in progress

3.2.2.27. GetExtCableStatus

GetExtCableStatus retrieves telemetry values PSU1_CABLE_OK and PSU2_CABLE_OK.

Table 3-38: Payload parameters for GetExtCableStatus

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultRebootBusy: Reboot sequence in progress
	bool	Cable status 1	PSU1_CABLE_OK
	bool	Cable status 2	PSU2_CABLE_OK

3.2.2.28. GetExtPsuStatus

GetExtPsuStatus retrieves telemetry values PSU1_DC_OK, PSU2_DC_OK, PSU1_TEMP_OK and PSU2_TEMP_OK.

Table 3-39: Payload parameters for GetExtPsuStatus

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultRebootBusy: Reboot sequence in progress
	bool	Psu dc good 1	PSU1_DC_OK value
	bool	Psu dc good 2	PSU2_DC_OK value
	bool	Psu temp 1	PSU1_TEMP_OK value
	bool	Psu temp 2	PSU2_TEMP_OK value

3.2.2.29. GetQcmIdTime

GetQcmIdTime retrieves telemetry values. When the lamp is on this value will be incremented and persisted by the YLCM application every 10 minutes as a 24 bit integer on the 1-wire EEPROM at address offset 0 (value may wrap-around).

Table 3-40: Payload parameters for GetQcmIdTime

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultLaserNotReady: Could not read out value from QCM ResultRebootBusy: Reboot sequence in progress
	uint64_t	Eeprom id	6 byte unique serial number within AT21CS11 (see [2.4])
	time	On time	`QCM_TOTAL_ON_TIME` value

3.2.2.30. GetLaserSerialNo

GetLaserSerialNo retrieves laser’s serial number over UART with “RSN” command (also see [2.3]).

Table 3-41: Payload parameters for GetLaserSerialNo

Payload	Type	Name	Description
In	-		

Out	uint8_t	Return value	ResultOk: No errors ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress ResultInvalidFormat: Laser returned “ERR: “ error code
	string	Serial number	Number returned as string by the laser unit

3.2.2.31. GetLaserCritErrorCode

GetLaserCritErrorCode retrieves laser’s critical error code over UART with “RMEC” command (also see [2.3]).

Table 3-42: Payload parameters for GetLaserCritErrorCode

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress ResultInvalidFormat: Laser returned “ERR: “ error code
	string	Critical error code	Critical error code returned as string by the laser unit

3.2.2.32. GetLaserCritErrorCount

GetLaserCritErrorCode retrieves laser’s critical error code over UART with “REC” command (also see [2.3]).

Table 3-43: Payload parameters for GetLaserCritErrorCount

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress ResultInvalidFormat: Laser returned “ERR: “ error code
	string	Critical error count	Critical error count returned as string by the laser unit

3.2.2.33. ResetLaserCritErrorCode

ResetLaserCritErrorCode attempts to clear laser’s critical error code over UART with the error code supplied by the user using “RCE” command (also see [2.3]).

Table 3-44: Payload parameters for ResetLaserCritErrorCode

Payload	Type	Name	Description
In	string	Critical error code	The critical error code to reset
Out	uint8_t	Return value	ResultOk: No errors ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress ResultInvalidFormat: Laser returned “ERR: “ error code

3.2.2.34. GetLaserSwVersion

GetLaserSwVersion retrieves laser’s software version over UART with “RFV” command (also see [2.3]).

Table 3-45: Payload parameters for GetLaserSwVersion

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress ResultInvalidFormat: Laser returned “ERR: “ error code
	string	Software version	Version returned as string by the laser unit

3.2.2.35. GetLaserStatus

GetLaserStatus retrieves laser's status over UART with "STA" command (also see [2.3]).

Table 3-46: Payload parameters for GetLaserStatus

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress ResultInvalidFormat: Laser returned "ERR: " error code
	uint32_t	Status	32bit status mask returned by the laser unit

3.2.2.36. GetLaserOnTime

GetLaserOnTime retrieves laser's internal enabled time over UART with "RET" command (also see [2.3]).

Table 3-47: Payload parameters for GetLaserOnTime

Payload	Type	Name	Description
In	-		
Out	uint8_t	Return value	ResultOk: No errors ResultLaserNotReady: Could not communicate with Laser ResultRebootBusy: Reboot sequence in progress ResultInvalidFormat: Laser returned "ERR: " error code
	time	On time	On time retrieved from laser

3.3. Firmware Upload

Every time the firmware is booted the YLCM will attempt to connect with via TFTP to a host server. The IP for this host server is obtained during retrieval of a DHCP address using additional meta-data. The TFTP server name is expected as an IP address string returned via option code 66 [2.2] (e.g. "192.168.192.200").

First a file (*lppsMk4/fw.info)* is retrieved which is used to check version numbering. The format of this file is the following data stored as plain text, where 'X' is a decimal digit:

"6681-18XX-XXXX
YLCM Firmware svnXXX"

- If version numbers match or are incompatible (e.g. lpps250 firmware detected), a log file of the boot process is uploaded to the server (*lppsMk4/fw.boot*) and starts the application.
- If version numbers do not match an upload of the actual firmware image (*lppsMk4/fw.img*) is executed and the device is rebooted afterwards.
- If TFTP is not available the YLCM will remain in the currently booted firmware and starts the application.

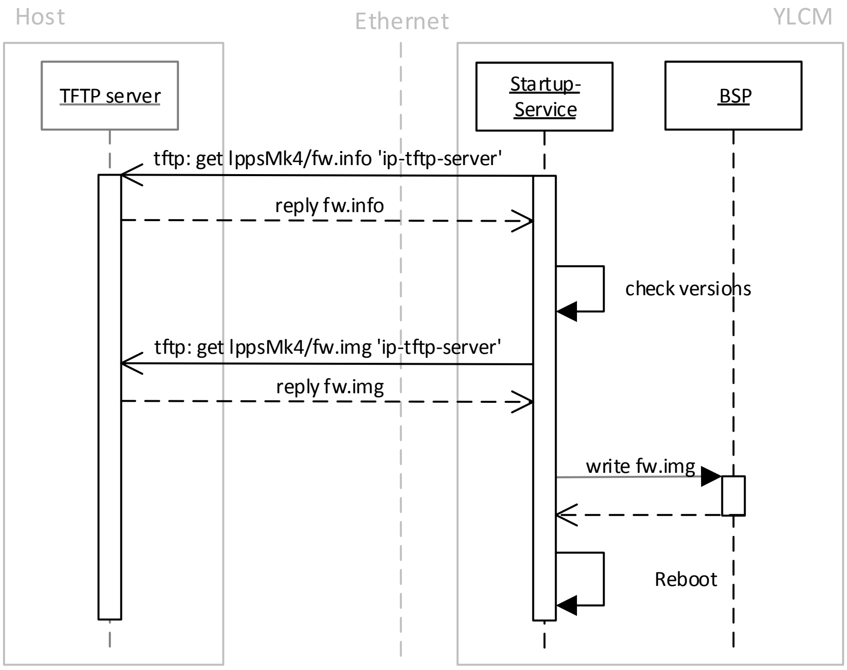
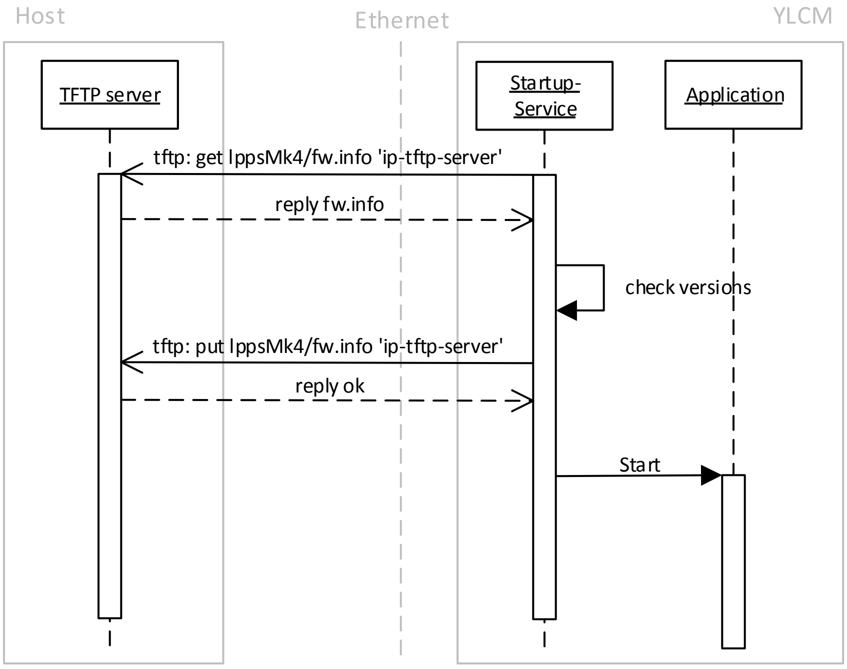


Figure 3-3: Firmware upload (version numbers differ) mechanism



5 *Figure 3-4: firmware upload (version numbers match) sequence*

4. YLCM Mk4 Test Application

Next to the command interface as described in section 3.2 a separate stand-alone YLCM Test Application can be used to monitor and control all functionality of the YLCM. This tool will comply with the following requirements.

4.1. Functional requirements

General:

- Connect via Ethernet (TCP/IP) to the YLCM Mk4. Upon connection to YLCM Mk1/2 an error will be generated due to incompatibility.
- Load/store all editable data fields via the GUI and green/red marks on telemetry into an initialization file
- Load a plain text file containing text strings for all strings (e.g. buttons, tabs, labels) used in the test application.
- 4 screens “Command interface”, “SW update”, “Trace Buffer”, and “Telemetry” providing functionalities as described below.

Command interface screen functionality:

- Call any of the Command Interface functions as defined in 3.2.2
- Store/Load total on time for lamp and pump laser in YLCM Mk4.
- Read/Write t_x & mode parameters for *SetLampOn* sequence (see 3.2.2.15)
- Read/Write *IrPowerOpticalThreshold* parameter (see Table 3-1)
- Read/Write *LampOpticalThreshold* parameter (see Table 3-1)
- Read/Write *IrPowerOpticalDisconnect* parameter (see Table 3-1)
- Read/Write *LampOpticalDisconnect* parameter (see Table 3-1)
- Read/Write *TmpQcmBlockThreshold* parameter (see Table 3-1)
- Read/Write *IrLaserControlGain* parameter (see Table 2-2)

SW update

- Upload firmware image
- Upload/download nvram data file
- Reboot controller
- Laser uart communication, with function to disable compatibility mode on laser via UART (“EDC” command)

Tracebuffer functionality:

- Read trace buffer from YLCM Mk4
- Clear trace buffer on-screen
- Store trace buffer to file
- Error messages will be marked with extra lines

Telemetry screen functionality:

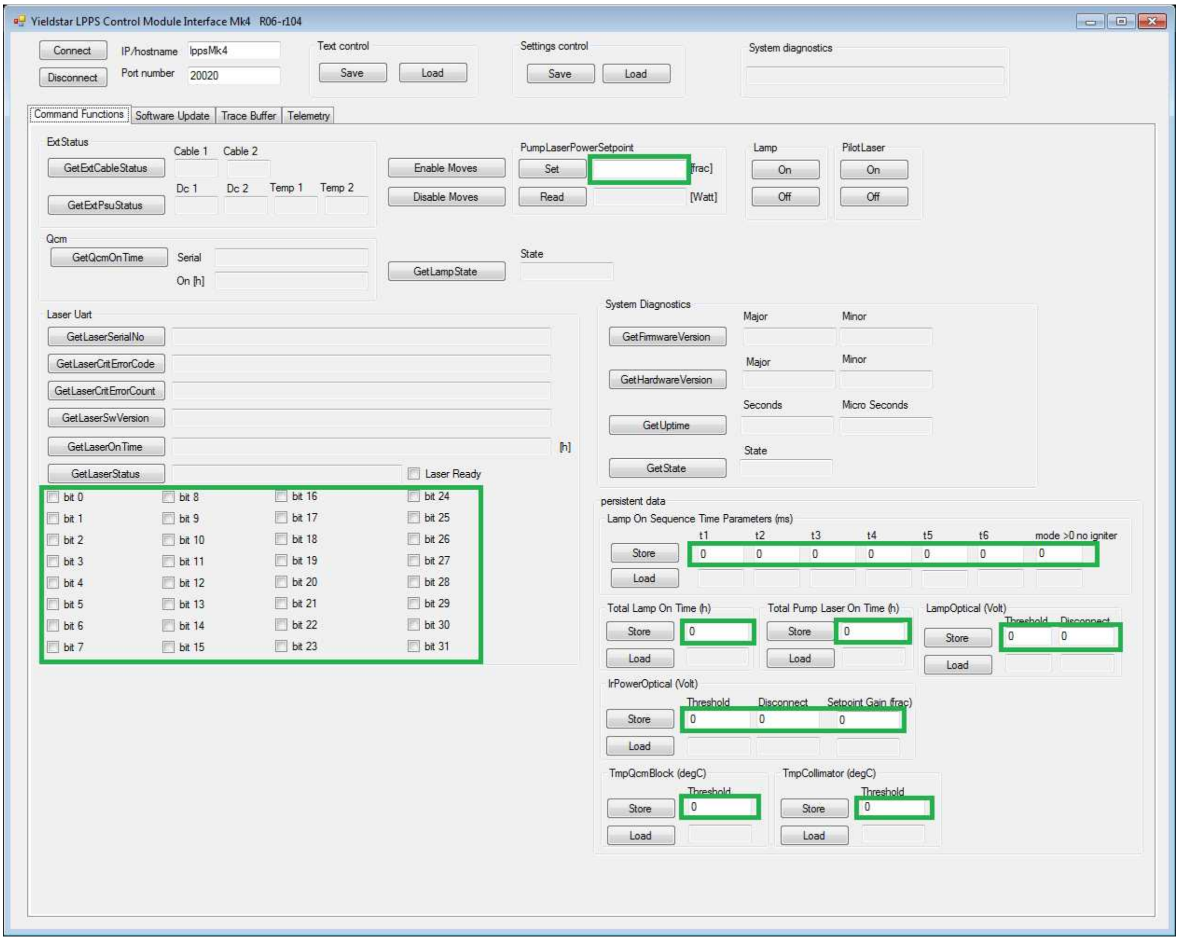
- Grid overview of all sensor & actuator data that can be retrieved including live updated values.
- Save all currently displayed values to CSV file

4.2. Non-functional requirements

- Stand-alone GUI based application
- Runs on Windows 7, 64-bit compatible OS
- Window height is limited to 768 pixels
- Tool comes with User Manual Document (UMD)

Below some examples are shown for indication of lay-out for the test application (***note: screens are subject to change and may appear different in the final product***).

-All edit/label boxes marked with **green** can be saved/loaded with preset values in settings XML file.
-All text labels can be saved/imported with preset values from a textual XML file.
-Upon startup the tool will attempt to ‘auto’ load text label XML files in the same directory as the location of the executable. This file must be named “ylcmmk4text.xml”



IP/hostname: lppsMk4
Port number: 20020

□ bit 0	□ bit 8	□ bit 16	□ bit 24
□ bit 1	□ bit 9	□ bit 17	□ bit 25
□ bit 2	□ bit 10	□ bit 18	□ bit 26
□ bit 3	□ bit 11	□ bit 19	□ bit 27
□ bit 4	□ bit 12	□ bit 20	□ bit 28
□ bit 5	□ bit 13	□ bit 21	□ bit 29
□ bit 6	□ bit 14	□ bit 22	□ bit 30
□ bit 7	□ bit 15	□ bit 23	□ bit 31

Lamp On Sequence Time Parameters (ms):

t1	t2	t3	t4	t5	t6	mode >0 no igniter
0	0	0	0	0	0	0

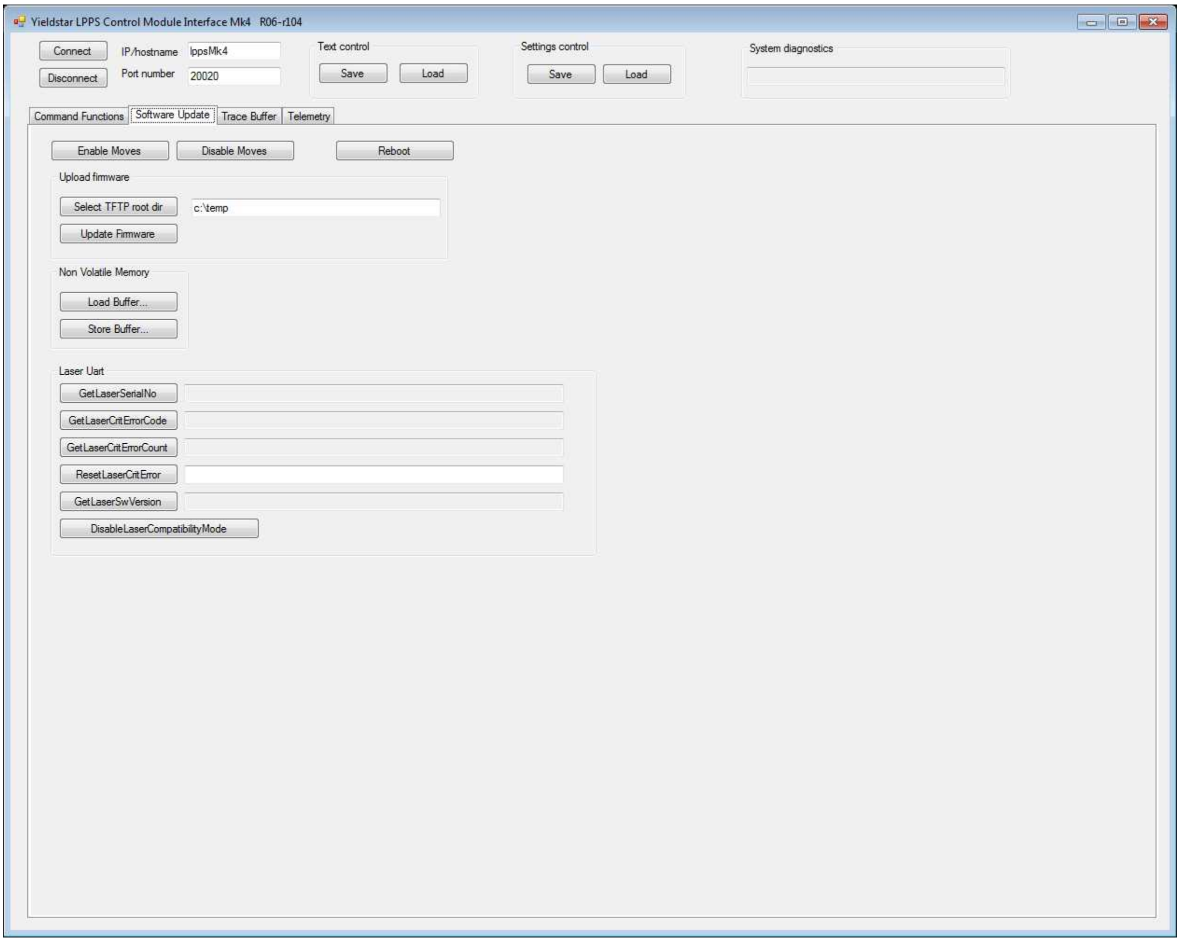
Total Lamp On Time (h): 0
Total Pump Laser On Time (h): 0

LampOptical (Volt): Threshold 0, Disconnect 0

IrPowerOptical (Volt): Threshold 0, Disconnect 0, Setpoint Gain (frac) 0

TmpQcmBlock (degC): Threshold 0
TmpCollimator (degC): Threshold 0

Figure 4-1: Example screenshot "Command functions"



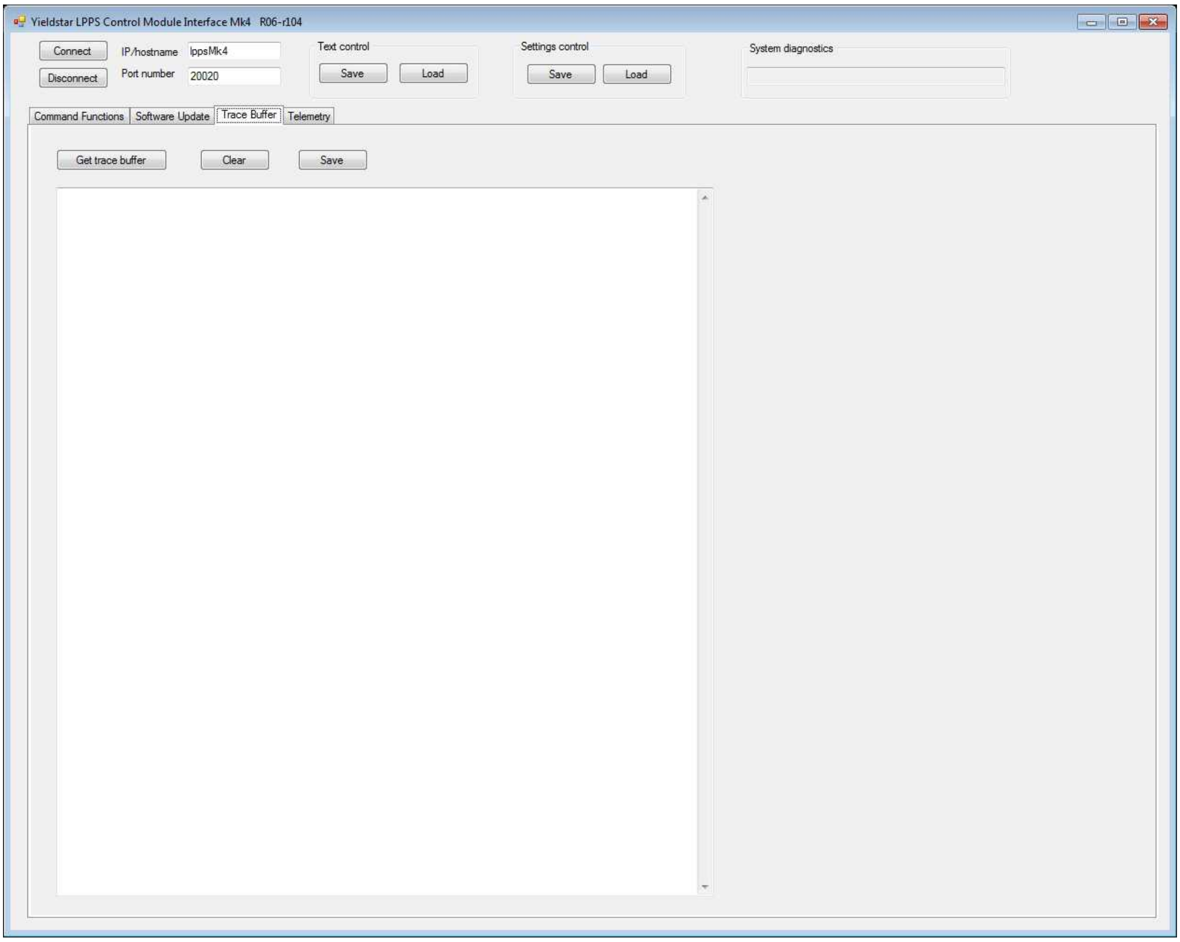
Yieldstar LPPS Control Module Interface Mk4 R06-r104

IP/hostname lppsMk4

Port number 20020

c:\temp

Figure 4-2: Example screen "Software update"

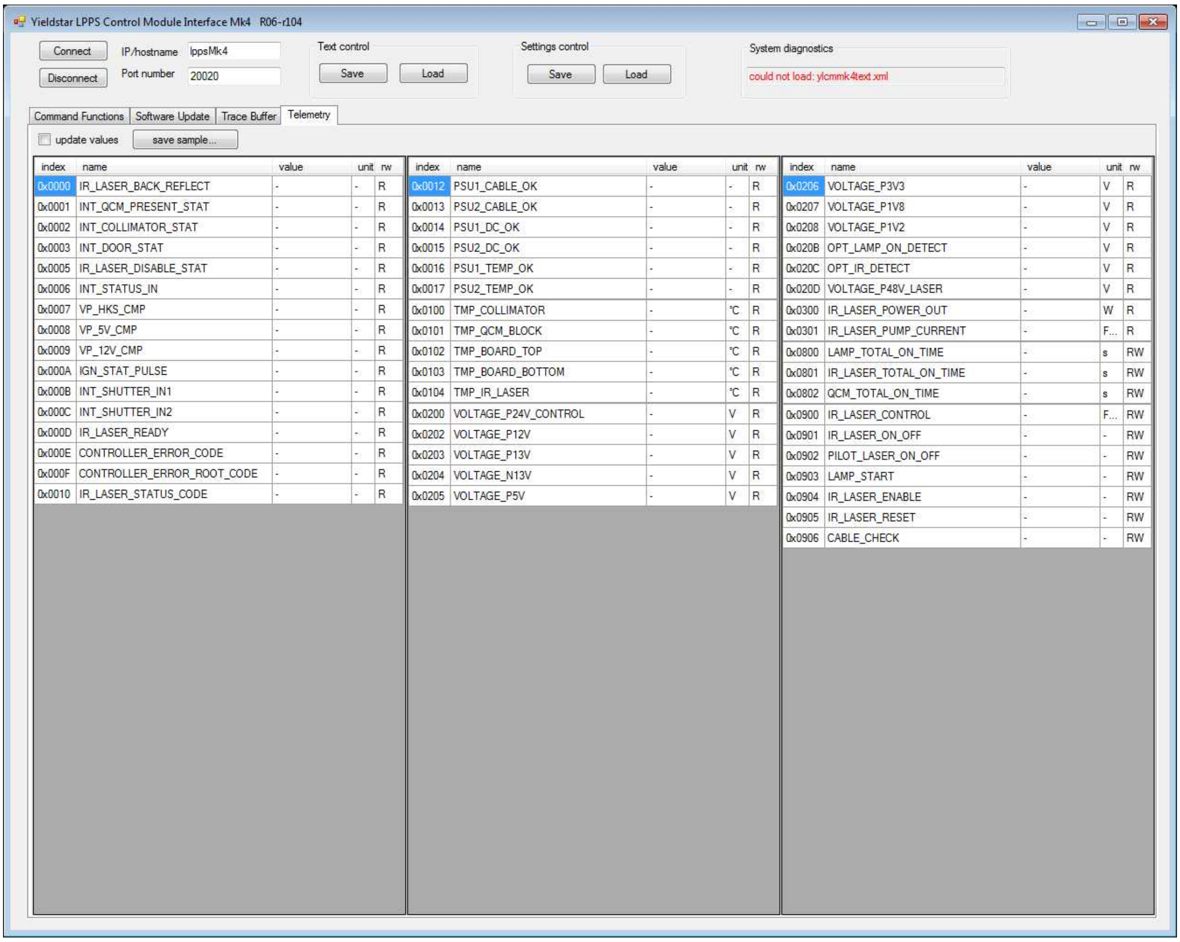


Yieldstar LPPS Control Module Interface Mk4 R06-r104

IP/hostname: lppsMk4

Port number: 20020

Figure 4-3: Example screen "Trace buffer interface"



IP/hostname lppsMk4

Port number 20020

could not load: ylcmmk4text.xml

□ update values

index	name	value	unit	rw
0x0000	IR_LASER_BACK_REFLECT	-	-	R
0x0001	INT_QCM_PRESENT_STAT	-	-	R
0x0002	INT_COLLIMATOR_STAT	-	-	R
0x0003	INT_DOOR_STAT	-	-	R
0x0005	IR_LASER_DISABLE_STAT	-	-	R
0x0006	INT_STATUS_IN	-	-	R
0x0007	VP_HKS_CMP	-	-	R
0x0008	VP_5V_CMP	-	-	R
0x0009	VP_12V_CMP	-	-	R
0x000A	IGN_STAT_PULSE	-	-	R
0x000B	INT_SHUTTER_IN1	-	-	R
0x000C	INT_SHUTTER_IN2	-	-	R
0x000D	IR_LASER_READY	-	-	R
0x000E	CONTROLLER_ERROR_CODE	-	-	R
0x000F	CONTROLLER_ERROR_ROOT_CODE	-	-	R
0x0010	IR_LASER_STATUS_CODE	-	-	R

index	name	value	unit	rw
0x0012	PSU1_CABLE_OK	-	-	R
0x0013	PSU2_CABLE_OK	-	-	R
0x0014	PSU1_DC_OK	-	-	R
0x0015	PSU2_DC_OK	-	-	R
0x0016	PSU1_TEMP_OK	-	-	R
0x0017	PSU2_TEMP_OK	-	-	R
0x0100	TMP_COLLIMATOR	-	°C	R
0x0101	TMP_QCM_BLOCK	-	°C	R
0x0102	TMP_BOARD_TOP	-	°C	R
0x0103	TMP_BOARD_BOTTOM	-	°C	R
0x0104	TMP_IR_LASER	-	°C	R
0x0200	VOLTAGE_P24V_CONTROL	-	V	R
0x0202	VOLTAGE_P12V	-	V	R
0x0203	VOLTAGE_P13V	-	V	R
0x0204	VOLTAGE_N13V	-	V	R
0x0205	VOLTAGE_P5V	-	V	R

index	name	value	unit	rw
0x0206	VOLTAGE_P3V3	-	V	R
0x0207	VOLTAGE_P1V8	-	V	R
0x0208	VOLTAGE_P1V2	-	V	R
0x020B	OPT_LAMP_ON_DETECT	-	V	R
0x020C	OPT_IR_DETECT	-	V	R
0x020D	VOLTAGE_P48V_LASER	-	V	R
0x0300	IR_LASER_POWER_OUT	-	W	R
0x0301	IR_LASER_PUMP_CURRENT	-	F...	R
0x0800	LAMP_TOTAL_ON_TIME	-	s	RW
0x0801	IR_LASER_TOTAL_ON_TIME	-	s	RW
0x0802	QCM_TOTAL_ON_TIME	-	s	RW
0x0900	IR_LASER_CONTROL	-	F...	RW
0x0901	IR_LASER_ON_OFF	-	-	RW
0x0902	PILOT_LASER_ON_OFF	-	-	RW
0x0903	LAMP_START	-	-	RW
0x0904	IR_LASER_ENABLE	-	-	RW
0x0905	IR_LASER_RESET	-	-	RW
0x0906	CABLE_CHECK	-	-	RW

Figure 4-4: Example screen "Telemetry interface"