	ПОВОРОТНО-НАКЛОННОЕ УСТРОЙСТВО
	Руководство по эксплуатации
Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

							C			
Перв. примен.	_						C	одержание		
	162.01	1. 0	Эписа	ние и работа (Системы.					4
	КПБМ.1162.	1.1 Назначение Системы и условия эксплуатации								4
Ĕ		1.2 Технические характеристики								5
		1	.3 Co	остав Системы	ſ					6
		. 1	.4 Уп	аковка Систем	иы					7
		2. \	Устро	йство и работа	а Систем	ы				8
		2	2.1 По	воротно-накло	онное уст	гройсті	во (ПНУ)		8
B. №		2	2.2 Бл	ок питания и с	ПЗ) игказ	[C)				11
Справ. №		2	2.3 Пу	льт управлени	я ПНУ					12
		3. I	Испол	ьзование Сист	гемы по н	назнач	ению			13
		3	.1 Эк	сплуатационн	ые огран	ичения	I			13
		3	.2 По	дготовка Сист	гемы к ис	спользо	ованию			13
3.3 Описание UDP протокола команд для управления ПНУ					14					
		3.3.1 Команда «Прочитать статус»							16	
дата		3.3.2 Команда «Двигаться в точку»						17		
одп. и дата		3.3.3 Команда «Загрузить точку»							17	
		3.3.4 Команда «Запуск последовательности»								18
<u>.</u>		3.3.5 Команда «Останов последовательности»							18	
Инв. № дубл		3.3.6 Команда «Задать максимальные ускорения и скорости»								19
Инв.			3.3.7	7 Команда «П <u>р</u>	очитать	макси	мальные	ускорения и скорости»		19
. №			3.3.8	3 Команда «За	дать смеі	цения	и ограни	ичения углов поворота»		20
Взам. инв. №										
Взе						[
æ										
Подп. и дата										
Подп										
		Изм.	Лист	№ докум.	Подп.	Дата				
		Разра	аб.	Емельянов					Лит. Лист Л	Тистов
10/1		Пров		Макаров			Пов	Onotho Harmonico	2	22
Инв. № подл.		Метр	олог	Журавлёв			1108	оротно-наклонное	ICTIA LID OO F	
ſÆ.		Н. кон	нтр.	Махиборода				устройство	КТИ НП СО Р	'AH
_	ı	\/ T D		Donyordan		1 1			Ī	

3.3.9 Команда «Прочитать смещения и ограничения углов поворота»	20
3.3.10 Команда «Сброс»	21
3.3.11 Команда «Изменить/прочитать IP адрес»	21
3.3.12 Команда «Изменить UDP порт»	22
3.4 Описание программы-утилиты для управления ПНУ через пульт	22
3.5 Действия в экстремальных ситуациях.	28
4. Техническое обслуживание	29
4.1 Порядок технического обслуживания	29
4.2 Монтаж и демонтаж	29
4.3 Регулировка и испытание Системы	29
5. Текущий ремонт	30
6. Меры безопасности при эксплуатации и ремонте	30
7. Хранение Системы	30
8. Транспортирование	30
9. Утилизация установки	31
10. Исходный код программы-утилиты ПУЛЬТА ПНУ	32

 Инв. № подл.
 Подп. и дата
 Взам. инв. №
 Инв. № дубл.
 Подп. и дата

Ізм.	Лист	№ докум.	Подп.	Дата

Лист 3 Руководство по эксплуатации содержит сведения о конструкции, принципе действия, области применения поворотно-наклонного устройства (далее — ПНУ), её составных частях и необходимые указания для правильной и безопасной эксплуатации.

К работе с ПНУ допускаются специалисты, владеющие навыками работы на персональном компьютере, прошедшие обучение под руководством разработчиков Системы и имеющие допуск по электробезопасности не ниже третьей группы.

В процессе эксплуатации Системы факторами опасности для обслуживающего персонала являются:

- подвижные механические части Системы;
- напряжение электропитания ~220 В.

1. ОПИСАНИЕ И РАБОТА СИСТЕМЫ.

1.1 Назначение Системы и условия эксплуатации.

- 1.1.1 Поворотно-наклонное устройство является несущим аппаратом, обеспечивающим возможность прецизионного управления позиционированием в горизонтальной и вертикальной плоскостях.
- 1.1.2 Внешнее управление поворотно-наклонным устройством, а так же информационное обеспечение осуществляется посредством команд, передаваемых по линии Ethernet протоколом UDP
- 1.1.3 Поворотно-наклонное устройство может управляться как самостоятельно от вынесенного пульта (миниатюрный компьютер/ноутбук с ПО управления), так от внешнего управляющего устройства (далее ВНУ).
- 1.1.4 Условия эксплуатации поворотно-наклонного устройства представлены в Таблице 1.

Изм. Лист № докум. Подп. Дата

ЛистarDelta

Подп. и дата

Инв. № дубл.

ž

Взам. инв.

Подп. и дата

1	Окружающая среда	Предназначено для работы в лабо-	
1		раторном помещении	
2	Окружающая эксплуатационная температура	От +10°C до +35°C	
3	Температура хранения	от -40°C до + 70°C	
4	Относительная влажность	93 % при 70°C	
5	Минимальное атмосферное давление	эксплуатационное 800 гПа	
3		устойчиво до 120 гПа	

1.2 Технические характеристики.

1.2.1 Технические параметры движения поворотно-наклонного устройства представлены в Таблице 2.

Таблица 2. Параметры движения

1	Пределы движения по азимуту	n x 360 ° (круговое)
2	Пределы движения по углу места	-95° до +95° (мах) (с установочной рамой для размещения изделия)
3	Пределы изменения скорости по азимуту	от 0,03 °/сек до 120 °/сек
4	Пределы изменения скорости по углу места	от 0,03 °/сек до 100 °/сек
5	Точность отработки заданного положения по азимуту и углу места	± 0,2 мрад
6	Точность повторяемости позиционирования по азимуту и углу места	± 0,2 мрад
7	Погрешность рассогласования положения по углу места относительно уровня параллельного основанию	± 2 ° (предварительная программная установка ± 1мрад)
8	Погрешность синхронизации по азимуту относительно вертикальной оси	± 2° (предварительная программная установка ± 1мрад)
9	Максимальная полезная нагрузка на оси	номинальная до 25 кг максимальная до 60кг
10	Максимальная статическая (несимметричная) нагрузка крутящего момента	45 нм
11	Максимальная полезная нагрузка инерционного крутящего момента	2 кгм ²
12	Максимальное азимутальное ускорение с сба- лансированной нагрузкой	12≥ рад/c ²
13	Максимальное ускорение по углу места с сбалансированной нагрузкой	12≥ рад/с ²

1.2.2 Технические параметры энергопотребления поворотно-наклонного устройства представлены в Таблице 3.

Таблица XXX. Параметры энергопотребления

1	Номинальное напряжение электропитания	24В постоянного тока
2	Допустимый диапазон электропитания	от 20В до 30В постоянного тока
3	Текущее значение потребляемого тока в неподвижном состоянии (при сбалансированной по-	≤ 0,8A

					_
					l
	-	3.0			
Изм.	Лист	№ докум.	Подп.	Дата	

Лист **5**

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Ī		лезной нагрузке без датчиков) U ном.= 24B	
	4	Среднее текущее значение потребляемого тока от источника электропитания (без запитки датчиков) при постоянной скорости движения 20 °/с и с сбалансированным номинальным полезным грузом, при температуре окр. среды при Т≥0°С и Uном= 24В	около 2А
	5	Текущее значение потребляемого тока во время старта с максимальным полезным грузом (без датчиков) при температуре окр. среды Т≥0°С и Uном= 24B	макс 11А в течение 0,5 сек

1.2.3 Технические параметры электрического интерфейса поворотно-наклонного устройства представлены в Таблице 4.

Таблица 4. Параметры электрического интерфейса

1	Информационно-управляющая сеть	Ethernet 10Mbit, протокол UDP
2	Максимальный расход энергии во время элек-	250W
	тропитания системы 24В	
	Источник для электропитания системы	Источник:
3	-	Вход: однофазн, ~ 230В, 50Гц
		Выход: =24B/14.6A/350W

1.2.4 Прочие требования на поворотно-наклонное устройство представлены в Таблице 5.

Таблица 5. Прочие требования

1	Ориентировочные габариты: Ш х В х Д	240 х 340 х 210 мм		
2	Вес поворотно-наклонного устройства без дат-	не более 20 кг		
	чиков			
3	Уровень защиты	IP 65 в соответствии с IEC 529		
4	Электромагнитная совместимость	Согласно MIL-CTD-461E		
5	Охлаждение	Естественное		
6	Защита	от перегрузки по току		
U		от перегрузки по напряжению		
7	Средний период между ремонтами	2 года в случае номинальной		
,		нагрузки		
8	Срок службы	10 лет в случае номинальной		
0		нагрузки		

1.3 Состав Системы.

1.3.1 Состав Системы приведен в таблице 6.

Таблица 6

Подп. и дата

Инв. № дубл.

ષ્ટ્ર

Взам. инв.

Подп. и дата

Инв. № подл.

Обозначение	Наименование	Кол.	Примечание

Изм.	Лист	№ докум.	Подп.	Дата	

6

Лист

Обозначение	Наименование	Кол.	Примечание
	Поворотно-наклонное устройство (ПНУ)	2	Первоначальный IP адрес: 192.168.6.1 и 192.168.6.2
	(DHC)		Порт: 10000
	Блок питания и связи (БПС)	2	
	Кабель питания и передачи данных	2	
	Патч-корд	3	1шт — для подключения пульта. 2шт - для двух внешних устройств управления. при подключении более одного внешнего устройства управления к одному ПНУ необходимо использовать дополнительный Ethernet switch.
	Пульт управления	1	Малогабаритный ноутбук с Ethernet интерфейсом и предустановленным на нем программным обеспечением
	Руководство по эксплуатации	1	

1.4 Упаковка Системы.

Перед транспортировкой или хранением наклонно-поворотное устройство размещается в деревянном коробе достаточного размера. Короб обеспечивает механическую защищенность во время транспортировки.

Изм.	Лист	№ докум.	Подп.	Дата

_{Лист}
7

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

2. УСТРОЙСТВО И РАБОТА СИСТЕМЫ.

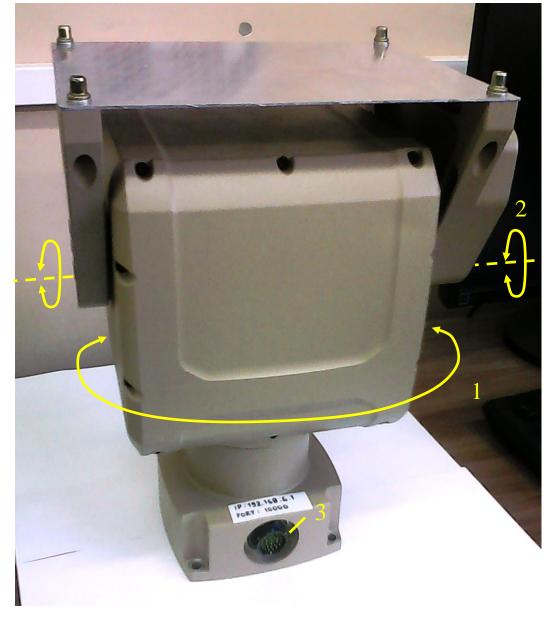
2.1 Поворотно-наклонное устройство (ПНУ).

2.1.1 Общее устройство ПНУ.

ПНУ изображено на рисунке 1. ПНУ предназначено для выполнения операций углового позиционирования тестируемого объекта и состоит из двух независимо управляемых угловых позиционеров (осей вращения):

- 1. Азимутальное позиционирование (ось азимута) позволяет выполнять повороты в горизонтальной плоскости, включая непрерывное вращение.
- 2. Наклонное позиционирование (ось наклона) позволяет выполнять угловые повороты относительно вертикальной оси.
- 3. Разъем предназначен для подачи силового питания и обеспечения Ethernet соединения с устройством.

Рис 1. Поворотно-наклонное устройство



Изм. Лист № докум. Подп. Дата

Подп. и дата

Инв. № дубл.

ષ્ટ્ર

Взам. инв.

Подп. и дата

Инв. № подл.

Лист

Упрощенная блок-схема ПНУ представлена на рис. 2 ниже:

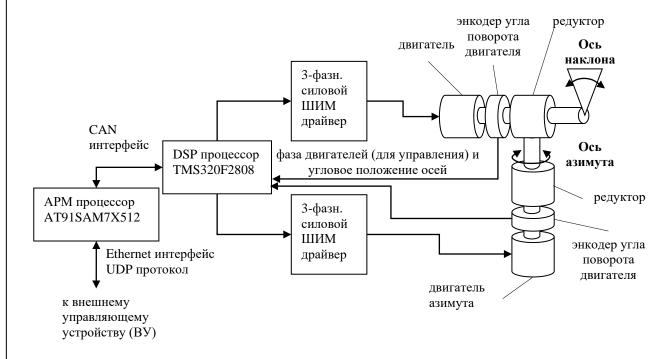


Рис. 2 Упрощенная блок-схема поворотно-наклонного устройства.

2.1.2 Управление движением ПНУ.

Подп. и дата

№ дубл.

Инв.

ષ્ટ્ર

Взам. инв.

дата

Подп. и

№ подл.

Оси поворота ПНУ (азимута или наклона) могут управляться внешним управляющим устройством (через UDP протокол) двумя способами: асинхронными и синхронным:

- При асинхронном управлении, внешнее управляющее устройство ВУ в любой момент может передать ПНУ целевые угловые координаты (углы поворота: азимут, наклон). Через некоторое (негарантированное) время, зависимости от скорости работы UDP протокола, команда будет доставлена в ПНУ. Полученное целевое положение немедленно принимается к исполнению в ПНУ. Такой метод вполне применим для системы, в которой пусть и малые, но негарантированные временные задержки UDP и операционной системы ВУ являются допустимыми. Этот метод более прост в программной реализации со стороны внешнего управляющего устройства ВУ.
- При синхронном управлении, внешнее управляющее устройство ВУ заблаговременно загружает буфер последовательных положений, и затем дает команду на выполнение значений из буфера. ПНУ последовательно исполняет команды в буфера с точным периодом в 10мс до опустошения буфера. Если необходима непрерывная работа в данном режиме, ВУ пополняет значения в буфере по мере его опустошения, для поддержания неразрывности потока данных. Несмотря на повышенную сложность

Изм.	Лист	№ докум.	Подп.	Дата

Лист **9**

программной реализации, такой метод позволяет создать более аккуратные траектории перемещения во времени.

Схематически оба способа управления изображена на рис. 3



Рис 3. Способы потокового управления положением ПНУ.

Внутри ПНУ по получении целевых координат на перемещение, управление двигателями, задающими углы поворота выполняется через алгоритм оптимального управления с целью минимизации времени исполнения команды.

Критерием оптимальности алгоритма является такой способ перемещения из произвольного текущего углового положения в заданное, при котором выполняются следующие условия:

- время перемещения из текущего угла в заданный минимально возможное (ограничено мощностью и динамическими характеристиками привода), с учетом всех остальных нижеперечисленных условий.
- пиковая скорость перемещения ограничена сверху параметром максимальной скорости.
- рабочее ускорение при перемещении ограничено сверху параметром максимального ускорения.
- конечная угловая скорость позиционирования в целевой точке равна нулю, (при условии, что ускорение не нулевое).
- алгоритм работает в режиме слежения: появление новой целевой угловой координаты даже при незаконченном позиционировании приводит к не-

Изм	Лист	№ докум.	Подп.	Дата	
Копиг	овап				

Подп. и дата

№ дубл.

Инв.

ષ્ટ્ર

Взам. инв.

дата

Подп. и

Инв. № подл.

Лист 10 медленному перерасчету и исполнению траектории с учетом текущей скорости и положения.

Введение такого алгоритма гарантирует минимизацию времени задержки при позиционировании при управляемых ограничениях скорости и ускорения.

Другими словами:

- время реакции ПНУ на целевое воздействие минимально (<1 мс)
- сам алгоритм перемещения выполнит перемещение за минимальное время (оптимальное управление с релейным ускорением).
- однако сама динамика движения осей поворота будет зависеть от ограничений максимального ускорения и скорости.

Поэтому выбор оптимальных ограничений ускорения и скорости является ключевым и должен подбираться под каждое конкретное применение.

Например:

Подп. и дата

Инв. № дубл.

ž

Взам. инв.

Подп. и дата

Инв. № подл.

- При большом заданном максимальном ускорении позиционирование из точки в точку выполняется с постоянной скоростью, равной параметру максимальной скорости, поскольку ускорение и торможение происходит очень быстро. Такой режим подходит для позиционирования, в котором время перемещения в целевое положение должно выполняться с минимальной временной задержкой. Однако, такой режим опасен для ПНУ с большой инерциальной массой, так как создает большие нагрузки (моменты вращения, ускорения) на его механическую часть и на силовые электронные компоненты. Возможны перегрузки по току (обратимые благодаря встроенной защите). Но также возможен повышенный износ (не исключается также повреждение) редуктора и подшипников осей. Поэтому такой режим имеет смысл только при малых массах.
- При малом максимальном ускорении минимизируются моменты и «удары», соотвествующие разгонам и торможению, а также «интерполируются перемещения с редким заданием координаты, что позволяет сглаживать движения. Этот режим удобен для плавных перемещений, включая большие массы. Такой режим работы более предпочтителен для долгосрочной работы ПНУ.

2.2 Блок питания и связи (БПС).

Блок питания и связи предназначен для обеспечения силового электроснабжения $\Pi H Y \ (+24B,\ 350Bt)$, а также обеспечивает Ethernet соединение внешнего устройства управления с $\Pi H Y$ через разъем RJ-45.

Изм.	Лист	№ докум.	Подп.	Дата	

Лист 11

Ниже приведен внешний вид лицевой и задней стороны БПС с описанием разъемов и элементов управления:



Разъем для подключения внешнего питания ~220B, 50Гц

Выключатель питания БПС



Ethernet – разъем для подключения к внешнему управляющему устройству

Силовой разъем для подключения ПНУ через кабель питания и передачи данных.

2.3 Пульт управления ПНУ.

Подп. и дата

Инв. № дубл.

ž

Взам. инв.

Подп. и дата

Инв. № подл.

- 2.3.1 Пультом управления является компактный РС-совместимый компьютер (мини-ноутбук), имеющий сетевую карту для Ethernet соединения и необходимое программное обеспечение для выполнение команд управления ПНУ по UDP интерфейсу. Программное обеспечение пульта будет описано ниже.
 - 2.3.2 Пульт управления подключается к БПС через Ethernet патчкорд.

Изм.	Лист	№ докум.	Подп.	Дата	

Лист 12

3. ИСПОЛЬЗОВАНИЕ СИСТЕМЫ ПО НАЗНАЧЕНИЮ

3.1 Эксплуатационные ограничения

- 3.1.1 Не допускается позиционировать изделие, масса которого превышает 25 кг. Потенциально, механика способна выполнить позиционирование нагрузок до 65кГ, однако все зависит от плеча, которое влияет на момент удержания нагрузки, которое должно быть минимизированным.
- 3.1.2 Не допускаются программные установки, ограничивающие максимальные ускорения и скорости, которые бы вызывали механические нагрузки на исполнительный механизм, превышающие его эксплуатационные параметры.
- 3.1.3 Следует обратить внимание, что при выключенном ПНУ, момент удержания его сильно ограничен, поэтому при больших отклонениях оси наклона от вертикали и большой массе изделия возможно самопроизвольное перемещение и удар. В связи с этим, при работе с большими полезными массами нагрузки не допускается оставлять нагрузку на стенде при больших углах отклонения от вертикали. В этом случае, после завершения работы, лучше всего демонтировать нагрузку или (если демонтаж затруднен или невозможен), то оставить наклон оси в вертикальном положении и ограничить опасное самопроизвольное опрокидывание оси.

3.2 Подготовка Системы к использованию

3.2.1 Распаковать Систему.

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

- 3.2.2 Произвести внешний осмотр. Убедиться в отсутствии повреждений (царапины, вмятины, и т.д.) .
- 3.2.3 Установить Систему в рабочее положение.
- 3.2.4 Выполнить кабельное соединение БПС с ПНУ.
- 3.2.5 Подключить внешнее устройство и/или прилагаемый пульт ПНУ через Ethernet соединение в БПС. При одновременном использовании ВУ и пульта ПНУ, необходимо использовать Ethernet switch.
- 3.2.6 Подключить БПС к сети переменного тока 220В, 5Гц.
- 3.2.7 Включить тумблер питания БПС.

Изм.	Лист	№ докум.	Подп.	Дата	

_{Лист}

- 3.2.8 Установить UDP соединение между внешним устройством управления и ПНУ. По умолчанию, параметры UDP соединения для ПНУ приведены здесь: IP адрес: 192.168.6.1 или 192.168.6.2, порт 10000
- 3.2.9 Выполнить команду протокола «Прочитать состояние», чтобы убедиться в наличии работающего UDP соединения.
- 3.2.10. Выполнить команду UDP протокола «Сброс», чтобы выполнить инициализацию двигателей ПНУ.
- 3.2.11. С этого момента ПНУ готово к работе с использованием команд UDP протокола.

3.3 Описание UDP протокола команд для управления ПНУ.

No

Инв. № подл.

Команда ВНУ

Все команды управления ПНУ, необходимые для внешним управляющим устройством (ВУ) вкратце перечислены в таблице ниже, далее будет приведено детальное описание команды, для того, чтобы программист внешнего управляющего устройства мог их применить в своем программном обеспечении:

Краткое функциональное описание

	U 1.=	110.11111111111111111111111111111111111	
	1.	«Прочитать статус»	ПНУ: возвращает текущее состояние,
			для анализа пользователем системы
	2.	«Двигаться в точку»	ВУ отправляет «точку» (значения
			азимутального и вертикального уг-
			лов).
<u>0</u>			
тда.			ПНУ: немедленно и асинхронно (с
Подп. и дата			минимальной латентностью) запус-
d			кает позиционирование точку.
	3.	«Загрузить точку»	ВУ: отправляет «точку» (значения
			азимутального и вертикального уг-
/бл.			лов).
і́е ду			
Инв. № дубл.			ПНУ: добавляет заданные значения
ИЕ			азимутального и вертикального углов
ુ			(«точку») во внутренний буфер для
В. У			дальнейшего синхронного позицио-
. ин			нирования.
Взам. инв. №	4.	«Запуск последовательности»	ПНУ: Предварительно загруженная
В			во внутренний буфер последователь-
			ность из «точек», запускается на ис-
та			полнение (позиционирование) с пе-
Подп. и дата			риодичностью 10мс.
[H]			
Под			По мере опустошения внутреннего
			буфера «точек», выполнение коман-

Изм. Лист № докум. Подп. Дата Копировал

Лист

		ды «Загрузить точку» позволяет по-
<u> </u>	, O a mayor ma a marina a mari	полнить буфер.
5.	«Останов последовательности»	ПНУ:
		Выполняет останов выполнения бу-
		фера «точек», если он был запущен.
6.	(20 HOW) MONOTH WITH WATER OF THE	Выполняет очистку буфера точек. ВУ:
0.	«Задать максимальные ускорения	
	и скорости»	Передает максимальные пиковые скорости и рабочие ускорения, зада-
		ваемые независимо для осей азимута
		и наклона.
		Эти параметры являются достаточно
		общими и позволяют задать разнооб-
		разные режимы позиционирования.
7.	«Прочитать максимальные уско-	ПНУ:
, .	рения и скорости»	Возвращает текущие значения пико-
	рения и екоростии	вых скоростей и рабочих ускорений
		для осей азимута и наклона
8.	«Задать смещения и ограничения	ВУ:
.	углов поворота»	Задает постоянные смещения осей
	Justob Hobopotum	при позиционировании, а также раз-
		решаемые пределы позиционирова-
		ния для оси наклона.
9.	«Прочитать смещения и ограни-	ВУ:
•	чения углов поворота»	Задает постоянные смещения осей
	Telling Jislob Hobepo iu	при позиционировании, а также раз-
		решаемые пределы позиционирова-
		ния для оси наклона.
10.	«Сброс»	ПНУ:
_ • •		Выполняет инициализацию внутрен-
		него состояния ПНУ.
		Оси позиционируются в исходное
		нулевое положение.
		Все параметры ПНУ принимают зна-
		чения по умолчанию. (за исключени-
		ем IP адреса и UDP порта)
11.	«Изменить IP адрес»	Позволяет устройству поменять соб-
	_	ственный IP адрес. Внимание! Де-
		лать это надо очень осторожно, с
		пониманием дела, иначе можно по-
		терять соединение с ПНУ
12.	«Изменить UDP порт»	Позволяет устройству поменять но-
		мер рабочего UDP порта. Внимание!
		Делать это надо очень осторожно, с
		пониманием дела, иначе можно по-
		терять соединение с ПНУ

Взам. инв. № | Инв. № дубл.

Подп. и дата

Инв. № подл.

```
3.3.1 Команда «Прочитать статус»
 ВУ отправляет в ПНУ команду:
 typedef struct {
    unsigned short
                     Command;
                                       код команды 0х0100
    unsigned short
                    nPacket;
                                       порядковый номер пакета*
 Размер структуры = 4 байт
 *ОПУ отслеживает номер пакета и в случае его потери взводиться соответствующий бит (см. далее). ОПУ
 ожидает увеличения номера пакета на 1 в каждой последующей команде. Для каждого узла используется своя
 цепочка порядковых номеров пакета. Первоначальное значение =1.
 Ответ на команду, принимаемый ВУ:
 typedef struct {
    unsigned short
                       Reply;
                                       код ответа 0х8100
    unsigned short
                       nPacket;
                                       порядковый номер пакета увеличивается на 1 в каждом, отправляе-
 мом узлом, пакете, после подачи питания и выполнения команды «Сброс» первый пакет имеет номер = 1.
    unsigned short
                       Error;
                                       Поле ошибок узла управления поворотами, наклонами (см. ниже)
    unsigned short
                       nPointBuf:
                                       Количество точек перемещения в буфере (0 - 255)
    unsigned short
                       MoveUnitState:
                                              Текущее состояние узла управления поворотами, наклонами
 (см. далее)
    unsigned short
                       StatErrorA;
                                       Копия состояния оси азимута - зарезервировано
    unsigned short
                       CoordinateA;
                                       Текущая координата оси поворота
    signed short
                       SpeedA;
                                       Текущая скорость оси поворота
    unsigned short
                       StatErrorE;
                                       Копия состояния оси наклона - зарезервировано
    unsigned short
                        CoordinateE;
                                       Текущая координата оси наклонов
    signed short
                       SpeedE;
                                       Текущая скорость оси наклонов
    unsigned short
                        VccPwrAE;
                                       напряжение питания сервоконтролеров (ст. байт - повороты)*
 Размер структуры = 24 байт
 *для расчета напряжения питания сервоконтроллеров необходимо воспользоваться след. формулой
 Напряжение питания (узла поворота)= ((VccPwrAE>>8)*30)/255;
 Напряжение питания (узла наклона)= ((VccPwrAE&0x00ff)*30)/255;
 Расшифровка поля ошибок узла управления поворотами, наклонами (unsigned short Error)
 D0 – ошибка формата или неизвестная команда
 D1 – была зафиксирована потеря пакета, сбрасывается командой «Прочитать состояние»
 D2 – ошибка в команде «Загрузить точку» (во всех режимах).
 D3 – нарушена последовательность точек в команде «Загрузить точку, режим «Слежение»»*
 D4 – ошибка в команде «Загрузить параметры узла управления поворотами, наклонами»
 D5 – ошибка в команде «Сменить IP адрес или № UDP порта »
 D6 – буфер точек перемещения пуст (была попытка чтения из пустого буфера)
 D7 – буфер точек перемещения полон (была попытка записи в полный буфер)
 D8 – ошибка внутренней шины CAN
 D9 – сервоконтролер оси поворота не работает
 D10 – сервоконтролер оси наклона не работает
 D11- D15 - резерв
 *последовательность считается нарушенной, если порядковый номер загружаемой точки меньше порядкового
 номера предварительно загруженной точки. Если они равны значение предварительно загруженной точки
 переписывается значением загружаемой точки.
 Расшифровка поля текущего состояние узла управления поворотами, наклонами (unsigned short MoveUnit-
 D0 – сервоконтролеры оси поворота и оси наклона к работе готовы.
 D1 – сервоконтролеры оси поворота и оси наклона активны
                                                                                                       Лист
Изм. Лист
            № докум.
                         Подп.
                                 Дата
```

Инв. № дубл.

윋

Взам. инв.

Подп. и дата

Инв. № подл.

```
D2 – резерв
 D3 – узел осуществляет перемещение в требуемую точку», в режиме «Позиция» или набор скорости, в режиме
 D4- Установлен режим работы «Позиция»
 D5- Установлен режим работы «Скорость»
 D6 – Установлен режим работы «Слежение»
 D7 – D15 - резерв
 Значения координат по углу поворота и наклона представляют собой целое число без знака (16 бит)
 Значение младшего разряда = 100 микрорадиан. Диапазон изменения по углу поворота 0 ... 62831
 (2*Pi*10000), по углу наклона устанавливается соответствующей командой (см далее).
 После подачи питания ОПУ устанавливается в точку с координатой 0 по углу поворота и 31415 (Рі *10000) по
 углу наклона. Значения скорости по углу поворота и наклона представляют собой целое знаковое число.
 Значение младшего разряда = 100 микрорадиан/сек.
 3.3.2 Команда «Двигаться в точку»
 Эта команда предназначена для асинхронных перемещений.
 ВУ посылает команду в ПНУ:
 typedef struct {
                                      код команды 0х0101
    unsigned short
                    Command;
                    nPacket;
    unsigned short
                                      порядковый номер пакета
    unsigned short
                    Azimuth;
                                      координата для поворота в горизонтальной плоскости
    unsigned short
                    Elevator;
                                      координата для поворота в вертикальной плоскости
 Размер структуры = 8 байт
 После получения команды начинается перемещение в точку с заданным значением координат, после дости-
 жения цели узел управления поворотами, наклонами отправляет следующий ответ.
 typedef struct {
    unsigned short
                       Reply;
                                      код ответа 0х8101
    unsigned short
                       nPacket;
                                      порядковый номер пакета
                       Azimuth;
    unsigned short
                                      координата в горизонтальной плоскости
    unsigned short
                       Elevator;
                                      координата в вертикальной плоскости
 Размер структуры = 8 байт
 Если в команде обнаружены ошибки возвращается полное состояние узла аналогичное ответу на команду
 «Прочитать статус» с установленным битом D2. Ошибкой считается попытка выполнить команду при не рабо-
 тающих сервоконтролерах ( биты D0,D1 поля MoveUnitState = 0). Если значения координат выходят за преде-
 лы рабочего диапазона перемещений, принудительно устанавливаются предельные значения, бит обнаружен-
 ной ошибки в этом случае не устанавливается.
 3.3.3 Команда «Загрузить точку»
 Эта команда предназначена для загрузки точки в буфер при синхронных перемещениях
 ВУ посылает в ПНУ структуру:
 typedef struct {
    unsigned short
                       Command;
                                              код команды 0х0102
    unsigned short
                       nPacket;
                                              порядковый номер пакета
    unsigned short
                       Azimuth;
                                              координата для поворота в горизонтальной плоскости
    unsigned short
                       Elevator;
                                              координата для поворота в вертикальной плоскости
    unsigned short
                       nPointMoveLoad;
                                              порядковый номер загружаемой точки *
 Размер структуры = 10 байт
                                                                                                      Лист
Изм. Лист
            № докум.
                         Подп.
                                 Дата
```

Инв. № дубл.

ષ્ટ્ર

Взам. инв.

Подп. и дата

Инв. № подл.

*загрузка точек начинается с порядкового номера =1, каждая последующая точка имеет порядковый номер больше на 1. Это необходимо для линейной интерполяции, если какие-нибудь точки будут потеряны (необходимо знать, сколько потеряно).

После загрузки первой точки ОПУ перемещается на требуемую позицию и ждет окончания перемещения, все последующие значения точек при этом, загружаются в буфер, затем, по окончанию перемещения, с периодом 10мс, из буфера начинают выбираться загруженные значения точек. Если буфер опустошается, ОПУ устанавливает бит D6 поля ошибок («Буфер пуст») и более ничего не делает. При появлении в буфере очередного значения ОПУ возобновляет перемещение. Так продолжается до смены режима работы. Если осуществляется загрузка значения точки в полностью заполненный буфер, устанавливается бит D7. Старое значение в буфере заменяется новым.

Ответ на команду имеет структуру:

Если в команде обнаружена ошибка возвращается полное состояние узла аналогичное ответу на команду «Прочитать статус» с установленным битами ошибок.

Ошибкой считается:

Инв. № подл.

Изм. Лист

№ докум.

Подп.

Дата

Попытка выполнить команду при не работающих сервоконтролерах (биты D0,D1 поля MoveUnitState = 0) в этом случае устанавливается бит D2. Нарушение последовательности точек, устанавливается бит D3.

Если загружаемые значения координат выходят за пределы рабочего диапазона перемещений, принудительно устанавливаются придельные значения, бит обнаруженной ошибки в этом случае не устанавливается. Если в команде ошибок не обнаружено ответ не возвращается.

3.3.4 Команда «Запуск последовательности»

```
ВУ посылает структуру:
         typedef struct {
Подп. и дата
            unsigned short
                                Command;
                                                // код команды 0х0110
            unsigned short
                                nPacket;
                                                // порядковый номер пакета
         Размер структуры = 4 байта
         При выполнении этой команды немедленно начинает проигрываться последовательность позиционирования
         по углу, загруженная во внутренний буфер. При успешном выполнении возвращается структура схожая с по-
Инв. № дубл.
         сылаемой:
         typedef struct {
            unsigned short
                                Command;
                                                // код команды 0х8110
            unsigned short
                                nPacket;
                                                // порядковый номер пакета
ત્ર
         Размер структуры = 4 байта
Взам. инв.
         3.3.5 Команда «Останов последовательности»
         ВУ посылает структуру:
         typedef struct {
дата
            unsigned short
                                Command;
                                                // код команды 0х0111
Подп. и
            unsigned short
                                nPacket;
                                                // порядковый номер пакета
         Размер структуры = 4 байта
```

```
При выполнении этой команды немедленно останавливается последовательность позиционирования по углу,
 загруженная во внутренний буфер.
 Неиспользованные отсчеты, хранящиеся в буфере сбрасываются.
 При успешном выполнении запуска возвращается структура схожая с посылаемой:
 typedef struct {
    unsigned short
                       Command;
                                      // код команды 0х8111
    unsigned short
                       nPacket:
                                      // порядковый номер пакета
 Размер структуры = 4 байта
 3.3.6 Команда «Задать максимальные ускорения и скорости»
 ВУ посылает структуру:
 typedef struct {
    unsigned short
                       Command:
                                              // код команды: 0х0140
    unsigned short
                       nPacket;
                                              // порядковый номер пакета
    unsigned short
                       MaxAccelAzimuth;
                                              // максимальное ускорение по азимуту (в горизонтальной
 плоскости)
    unsigned short
                       MaxAccelElevator;
                                              // максимальное ускорение по наклону (в вертикальной плос-
 кости)
    unsigned short
                       MaxVelocityAzimuth;
                                             // максимальная скорость по азимуту (в горизонтальной плос-
 кости)
    unsigned short
                       MaxVelocityElevator;
                                             // максимальная скорость по наклону (в вертикальной плос-
 кости)
 Размер структуры = 12 байт
 При выполнении этой команды в ПНУ задаются параметры максимальных скоростей и ускорений, определя-
 ющих динамику осей. При успешном выполнении возвращается структура схожая с посылаемой, возвращаю-
 щая текущие установки:
 typedef struct {
    unsigned short
                       Command;
                                              // код команды: 0х8140
    unsigned short
                       nPacket:
                                              // порядковый номер пакета
    unsigned short
                       MaxAccelAzimuth;
                                              // максимальное ускорение по азимуту (в горизонтальной
 плоскости)
    unsigned short
                       MaxAccelElevator;
                                              // максимальное ускорение по наклону (в вертикальной плос-
 кости)
    unsigned short
                       MaxVelocityAzimuth;
                                             // максимальная скорость по азимуту (в горизонтальной плос-
 кости)
    unsigned short
                       MaxVelocityElevator;
                                             // максимальная скорость по наклону (в вертикальной плос-
 кости)
 Размер структуры = 12 байт
 3.3.7 Команда «Прочитать максимальные ускорения и скорости»
 Команда аналогична команде «Задать максимальные ускорения и скорости», однако посылаемая структура
 урезана. ВУ посылает структуру:
 typedef struct {
    unsigned short
                       Command;
                                              // код команды: 0х0140
    unsigned short
                       nPacket;
                                              // порядковый номер пакета
 Размер структуры = 4 байта
                                                                                                      Лист
Изм. Лист
            № докум.
                         Подп.
                                 Дата
```

дата

Подп. и /

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп. и

№ подл.

```
При успешном выполнении возвращается структура, возвращающая текущие установки:
 typedef struct {
    unsigned short
                       Command;
                                              // код команды: 0х8140
    unsigned short
                       nPacket;
                                              // порядковый номер пакета
    unsigned short
                       MaxAccelAzimuth;
                                              // максимальное ускорение по азимуту (в горизонтальной
 плоскости)
    unsigned short
                       MaxAccelElevator;
                                              // максимальное ускорение по наклону (в вертикальной плос-
 кости)
    unsigned short
                       MaxVelocityAzimuth;
                                             // максимальная скорость по азимуту (в горизонтальной плос-
 кости)
                                              // максимальная скорость по наклону (в вертикальной плос-
    unsigned short
                       MaxVelocityElevator;
 кости)
 Размер структуры = 12 байт
 3.3.8 Команда «Задать смещения и ограничения углов поворота»
 ВУ посылает структуру:
 typedef struct {
    unsigned short
                       Command;
                                      // код команды: 0х0120
    unsigned short
                       nPacket;
                                      // порядковый номер пакета
    signed short
                       OffsetAzimuth; // смещение в горизонтальной плоскости
    signed short
                       OffsetElevator; // смещение в вертикальной плоскости
    unsigned short
                       MinElevator; // минимальная координата в вертикальной плоскости
    unsigned short
                       MaxElevator; // максимальная координата в вертикальной плоскости
 Размер структуры = 12 байт
 При выполнении этой команды в ПНУ задаются параметры ограничений и смещений углов поворота. При
 успешном выполнении запуска возвращается структура схожая с посылаемой, возвращающая текущие уста-
 новки:
 typedef struct {
    unsigned short
                       Command;
                                      // код команды: 0х8120
    unsigned short
                       nPacket;
                                      // порядковый номер пакета
    signed short
                       OffsetAzimuth; // смещение в горизонтальной плоскости
    signed short
                       OffsetElevator; // смещение в вертикальной плоскости
    unsigned short
                       MinElevator;
                                      // минимальная координата в вертикальной плоскости
    unsigned short
                       MaxElevator;
                                      // максимальная координата в вертикальной плоскости
 Размер структуры = 12 байт
 3.3.9 Команда «Прочитать смещения и ограничения углов поворота»
 ВУ посылает структуру аналогичную при задании ограничений и смещения углов поворота, только урезанную:
 typedef struct {
    unsigned short
                       Command:
                                      // код команды: 0х0120
    unsigned short
                       nPacket;
                                      // порядковый номер пакета
 Размер структуры = 4 байт
 При успешном выполнении возвращается структура, с текущими установками:
 typedef struct {
                                                                                                      Лист
Изм. Лист
            № докум.
                         Подп.
                                 Дата
```

. № дубл.

Инв.

윋

Взам. инв.

дата

Подп. и

Инв. № подл.

```
unsigned short
                        Command:
                                       // код команды: 0х8120
    unsigned short
                        nPacket:
                                       // порядковый номер пакета
    signed short
                        OffsetAzimuth; // смещение в горизонтальной плоскости
    signed short
                        OffsetElevator; // смещение в вертикальной плоскости
    unsigned short
                        MinElevator;
                                       // минимальная координата в вертикальной плоскости
    unsigned short
                        MaxElevator:
                                       // максимальная координата в вертикальной плоскости
 Размер структуры = 12 байт
 3.3.10 Команда «Сброс»
 При выполнении команды, ПНУ инициализирует внутреннее состояние.
 Все параметры ПНУ принимают значения по умолчанию. (за исключением IP адреса и UDP порта)
 ВУ посылает структуру в ПНУ:
 typedef struct {
    unsigned short
                        Command:
                                       // код команды 0х01F0
    unsigned short
                        nPacket;
                                       // порядковый номер пакета
 Размер структуры = 4байта
    Команда осуществляет сброс сервоконтролеров управления поворотами, наклонами. Осуществляет пере-
 мещение платформы в начальную позицию. Осуществляется сброс ошибок, очистку буфера точек перемеще-
 ния. Время выполнения команды может достигать 1 мин. В этот период подача любых команд не допустима.
    Если команда выполнена, возвращается след. ответ.
 typedef struct {
    unsigned short
                        Reply:
                                       // код ответа 0х81F0
    unsigned short
                        nPacket;
                                       // порядковый номер пакета = 0
 Размер структуры = 4 байта
 3.3.11 Команда «Изменить/прочитать IP адрес»
 При задании нового ІР адреса, ВУ посылает структуру в ПНУ:
 typedef struct {
    unsigned short
                        Command;
                                       // код команды = 0x0F00
    unsigned short
                        nPacket;
                                       // порядковый номер пакета
    unsigned char
                        IpAdr[4];
                                       // новый IP адрес
    unsigned char
                        CopyIpAdr[4]; // копия нового IP адреса
 Размер структуры = 12 байт
 *пример: Задание ІР адреса: 192.168.10.2 должно быть следующим образом:
 IpAdr[0] = 192;
 IpAdr[1] = 168;
 IpAdr[2] = 10;
 IpAdr[3] = 2;
 CopyIpAdr[0] = 192;
 CopyIpAdr[1] = 168;
 Copy IpAdr[2] = 10;
 CopyIpAdr[3] = 2;
 В случае корректного выполнения команды ПНУ возвращает следующий ответ на команду:
 typedef struct {
    unsigned short
                        Reply;
                                       // код ответа 0х8F00
    unsigned short
                        nPacket;
                                               // порядковый номер пакета
 Размер структуры = 4 байт
                                                                                                        Лист
Изм. Лист
            № докум.
                         Подп.
                                 Дата
```

Инв. № дубл.

윋

Взам. инв.

дата

Подп. и ,

Инв. № подл.

Если в команде обнаружены ошибки возвращается полное состояние узла аналогичное ответу на команду «Прочитать статус». С установленным битом D0 поля ошибок. Ошибкой считается не совпадение полей IpAdr и CopyIpAdr в команде. Ответ на команду, в любом случае, осуществляется со старым значением IP адреса. В случае корректного выполнения команды ОПУ после формирования ответа изменяет IP адрес, и все последующие команды ожидает с новым значением IP.

```
* * *
```

```
Если ВУ посылает запрос без параметров
typedef struct {
  unsigned short
                       Command:
                                      // код команды 0x0F00
  unsigned short
                       nPacket;
                                      // порядковый номер пакета
Размер структуры = 4 байт
Тогда ПНУ возвращает след. ответ.
typedef struct {
  unsigned short
                       Reply;
                                      // код ответа 0х8F00
  unsigned short
                       nPacket;
                                      // порядковый номер пакета
  unsigned char
                       MacAdr[6];
                                      // МАС адрес ОПУ
  unsigned char
                                      // ІР адрес ОПУ
                       IpAdr[4];
                       MovePort;
                                      // № UDP порта узла управления поворотами наклонами
  unsigned short
                       Com1Port;
                                      // № UDP порта последовательного интерфейса №1
  unsigned short
                       Com2Port;
  unsigned short
                                      // № UDP порта последовательного интерфейса №2
                       Com4Port;
  unsigned short
                                      // № UDP порта последовательного интерфейса №4
  unsigned short
                       InOutPort;
                                      // № UDP порта узла управления подачей питания на нагрузку
Размер структуры = 24 байта
```

3.3.12 Команда «Изменить UDP порт»

```
ВУ посылает структуру: typedef struct {
```

Подп. и дата

Инв. № дубл.

ત્ર

Взам. инв.

Подп. и дата

Инв. № подл.

```
unsigned short<br/>unsigned short<br/>unsigned short<br/>unsigned shortCommand;<br/>nPacket;<br/>UdpPort;// код команды = 0x01FF<br/>// порядковый номер пакета<br/>// новое значение порта<br/>CopyUdpPort;unsigned shortCopyUdpPort;// копия нового значения порта
```

Размер структуры = 8 байт

Если в команде обнаружены ошибки возвращается полное состояние узла аналогичное ответу на команду «Прочитать статус». С установленным битом D5 поля ошибок. Ошибкой считается не совпадение полей UdpPort и CopyUdpPort в команде, а также попытка установить номер, совпадающий с номером UDP порта другого узла ОПУ. Ответ на команду, в любом случае, осуществляется со старым значением UDP порта. В случае корректного выполнения команды узел после формирования ответа изменяет номер порта, и все последующие команды ожидает с новым значением номера.

3.4 Описание программы-утилиты для управления ПНУ через пульт.

Для целей тестирования протокола и настройки рабочих параметров ПНУ создана программа-утилита, реализующая все описанные здесь команды

Изм.	Лист	№ докум.	Подп.	Дата

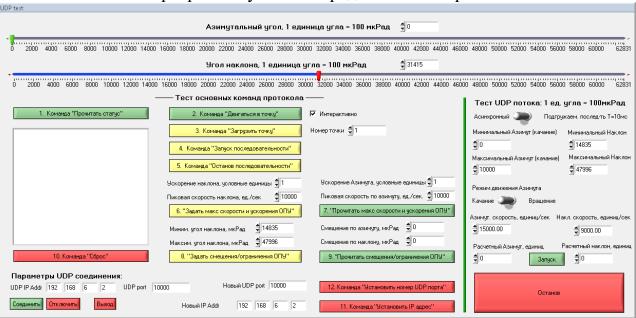
_{Лист}

UDP протокола (см. п. 3.3). Эта утилита установлена на пульте управления ПНУ.

Кроме того, в приложении, прилагается исходный текст утилиты на языке С, в среде разработки LabWindows, чтобы программист устройства внешнего управления мог позаимствовать элементы кода, которые могут послужить простыми примерами реализации его собственной программы.

Здесь описан графический интерфейс и руководство оператора по работе с утилитой пульта.

3.4.1 Главное окно программы-утилиты представлено на рис ниже:



3.4.2 Главное окно утилиты состоит из следующих функциональных блоков:

• «слайдеров» азимутального и наклонного углов для интерактивного задания координат:



• Кнопок и элементов для создания и выключения UDP соединения:



• Левой части окна для тестирования и ручной проверки/использования всех 12 команд протокола. Для каждой из команд имеется отдельная кнопка с номером команды соответствующим списку команд перечисленных в таблице в пункте 3.3

Ізм.	Лист	№ докум.	Подп.	Дата

_{Лист}

Подп. и дата

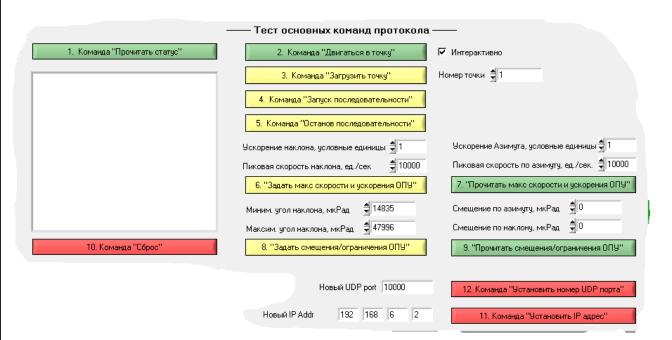
Инв. № дубл.

ષ્ટ્ર

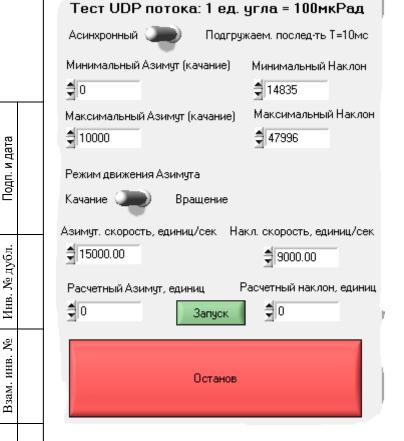
Взам. инв.

дата

Подп. и



• Небольшой программы для тестирования непрерывной загрузки последовательности и исполнения ее выполнения в синхронном или асинхронном режиме:



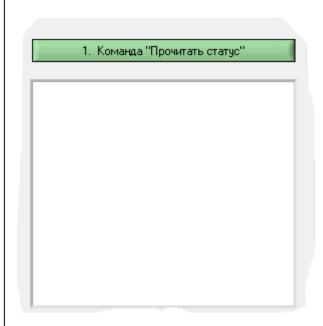
- 3.4.3 Краткие описания команд и элементов графического интерфейса:
- 3.4.3.1 Команда «1.Прочитать статус» имеет следующие элементы кнопку и окошко, в которое выводится полученный ответ:

Изм.	Лист	№ докум.	Подп.	Дата	

_{Лист} 24

дата

Подп. и



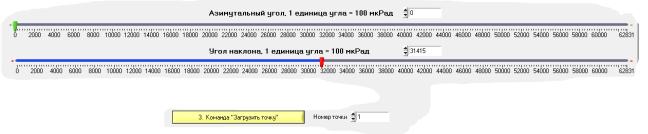
3.4.3.2 Команда «2. Двигаться в точку» имеет следующие рабочие элементы:



Нажатие кнопки «2. Команда Двигаться в точку» приводит к чтению углов азимута и наклона и вызывает немедленное исполнение этих команд позиционером.

Если в поле «Интерактивно» стоит «галочка», то изменение углов слайдеров немедленно выполняется (без нажатия кнопки «2. Команда Двигаться в точку»).

3.4.3.3 Команда «3. Загрузить точку» имеет следующие рабочие поля:



При нажатии кнопки, значение азимута и угла наклона добавляется в буфер ПНУ под номером точки указанном в соответствующем поле.

3.4.3.4 Команда запуска последовательности расположенной в буфере ПНУ

4. Команда "Запуск последовательности"

вызывается на жатием кнопки:

T.T.	т.	3.0	17	
Изм	Пист	№ локум	Полп	Пата

Лист 25

Подп. и дата

Инв. № дубл.

ž

Взам. инв.

Подп. и дата

ных скоростей и ускорений осей: Ускорение Азимута, условные единицы 🖣 1 Ускорение наклона, условные единицы 🗐 1 Пиковая скорость по азимуту, ед./сек. \$ 10000 **\$** 10000 Пиковая скорость наклона, ед./сек 6. "Задать макс скорости и ускорения ОПУ" Ускорение задается в условных единицах от 1 – минимальное, до 12 - максимальное. Пиковая скорость поворота задается в единицах угла (100мкРад) в секунду. 3.4.3.7 Чтение максимальных значений достигается нажатием соответствующей кнопки, которая обновляет значения соответствующих полей скорости и ускорения: Ускорение Азимута, условные единицы 🖣 1 Ускорение наклона, условные единицы Пиковая скорость по азимуту, ед./сек. 🖨 10000 **\$** 10000 Пиковая скорость наклона, ед./сек 7. "Прочитать макс скорости и ускорения ОПУ" 3.4.3.8 Нажатие указанной ниже кнопки приводит к тому, что соответствующие смещения и ограничения углов передаются в ПНУ: Подп. и дата Смещение по азимуту, мкРад Миним, угол наклона, мкРад Смещение по наклону, мкРад Максим, угол наклона, мкРад 8. "Задать смещения/ограничения ОПУ" № дубл. 3.4.3.9 Нажатие указанной ниже кнопки приводит к тому, что соответствую-Инв. щие смещения и ограничения углов читаются из ПНУ и отображаются в соответствующих значениям полях: ત્ર Взам. инв. Смещение по азимуту, мкРад Миним, угол наклона, мкРад Смещение по наклону, мкРад Максим, угол наклона, мкРад 9. "Прочитать смещения/ограничения ОПУ" 3.4.3.10 Нажатие кнопки «10. Сброс» вызывает инициализацию и позициони-Подп. и дата рование осей в исходное (нулевое) положение. Все параметры осей сбрасываются в значения по умолчанию. № подл. Лист Изм. Лист № докум. Подп. Дата Копировал Формат А4

3.4.3.5 Команда останова выполнения последовательности расположенной в

3.4.3.6 Нажатие кнопки и соответствующие поля задают пределы максималь-

вызывается

нажатием

кнопки:

ee

сброса

ПНУ

5. Команда "Останов последовательности"

буфере

10. Команда "Сброс"

3.4.3.11 Эти графические элементы позволяют поменять рабочий IP адрес Новый IP Addr 192 168 6 2 11. Команда "Установить IP адрес"

ПНУ:

3.4.3.12 Эти графические элементы управления позволяют поменять номер рабочего порта UDP соединения ПНУ:

Новый UDP port 10000 12. Команда "Установить номер UDP порта"

- 3.4.3.13 Этот переключатель задает режим подачи координат в ПНУ при потоковой подгрузке данных:

 Aсинхронный Подгружаем. послед-ть Т=10мс
 - Асинхронный режим немедленное исполнение команд позиционирования.
 - Синхронный режим использование буфера команд в ПНУ, который исполняется с периодичностью в 10мс.
- 3.4.3.14 При вычислении задаваемого угла наклона в потоке используются следующие поля:

Минимальный Наклон
₹ 14835
Максимальный Наклон
47996
Накл. скорость, единиц/сек
9000.00
Расчетный наклон, единиц
. 0

Подп. и дата

Инв. № дубл.

ž

Взам. инв.

Подп. и дата

Инв. № подл.

Наклон меняется с наклонной скоростью в единцах угла в секунду, например прирастает, и достигая максимального или минимального значений, скорость наклона меняет знак на противоположный. Тем самым происходит изменение координаты в режиме «пинг-понг» (от границы до границы, где скорость известна и может задаваться в процессе выполнения эксперимента).

• Расчетный наклон – это текущий угол наклона, а также его начальное значение при старте эксперимента.

Изм.	Лист	№ докум.	Подп.	Дата

_{Лист} 27

- Наклонная скорость это скорость изменения угла наклона в единицах угла в секунду.
- Максимальное и минимальное значения этопределы угла в которых он может изменяться.
- 3.4.3.15 Аналогично наклону, но независимо от него может вычисляться и задаваемый азимут:

Минимальный Азимут (качание)
₹0
Максимальный Азимут (качание) \$\frac{10000}{}\$
Режим движения Азимута
Качание Вращение
Азимут, скорость, единиц/сек
\$ 15000.00
Расчетный Азимут, единиц
\$ 0

В режиме задания азимута есть также опция «Вращение», в которой максимальное и минимальное значения азимута игнорируются и его текущее значение определяется только азимутальной скоростью.

3.4.3.16 Нажатие кнопок Запуск и Останов, соответственно запускает или останавливает эксперимент по потоковому заданию изменяющихся азимута и



3.5 Действия в экстремальных ситуациях.

- 3.5.1 При внезапном появлении признаков экстремальной ситуации (механический удар, пожар, задымление или опасное механическое поведение ПНУ) необходимо немедленно обесточить ПНУ выключением выключателя силового питания на панели БПС.
- $3.5.2~{\rm При}$ неотложной ситуации, когда доступ к выключателю затруднен резко выдернуть шнур питания 220В из гнезда питания БПС.

1_					
<u> </u>					
ı					
1_					
I.	Ізм.	Лист	№ докум.	Подп.	Дата

Лист 28

Подп. и дата

Инв. № дубл.

ž

Взам. инв.

Подп. и дата

3.5.3 При отключении Системы в экстремальных ситуациях повторное включение Системы осуществляется после устранения причин экстремальной ситуации.

4. ТЕХНИЧЕСКОЕ ОБСЛУЖИВАНИЕ

4.1 Порядок технического обслуживания

- 4.1.1 К работе с Системой допускаются лица, имеющие третью квалификационную группу по электробезопасности, изучившие настоящий документ и прошедшие инструктаж по технике безопасности.
- 4.1.2 Проверку технического состояния Системы следует осуществлять каждый раз перед началом работы в соответствии с требованиями

4.2 Монтаж и демонтаж

- 4.2.1 Монтаж Системы следует осуществлять непосредственно на жестком полу, или на жестких основаниях, которые могут обеспечить механические нагрузки, соответствующие применению ПНУ.
- 4.2.2 При демонтаже Системы все составные части уложить в транспортную тару.

4.3 Регулировка и испытание Системы

- 4.3.1 Регулировка ПНУ заключается в тестовом управлении ПНУ начинается при полном отсутствии механической нагрузки пользователя. Устанавливается только штатный поставляемый кронштейн. Начало работы следует начать только на малых ускорениях и скоростях перемещения, задаваемых при помощи прилагаемого пульта управления.
- 4.3.2 Постепенно изучаются пределы максимальных скоростей и ускорений, допустимых для задачи и подбираются их оптимальные значения без внешней нагрузки.
- 4.3.3 Устанавливается масса-эквивалент нагрузки, при которой снова плавно подбираются параметры максимальных скоростей и ускорений ПНУ.
- 4.3.4 После нахождения оптимальных режимов позиционирования, необходимые смещения углов, пределы позиционирования, максимальные скорости/ускорения, полученные на этом этапе должны быть учтены и автоматически подгружаться протоколом UDP программном обеспечении внешнего устройства управления ВУ.

Изм.	Лист	№ докум.	Подп.	Дата

_{Лист} 29

Подп. и дата

Инв. № дубл.

ષ્ટ્ર

Взам. инв.

Подп. и дата

5. ТЕКУЩИЙ РЕМОНТ

- 5.1 В течение гарантийного срока ремонт Системы безвозмездно выполняет предприятие изготовитель, а составных частей фирмы-изготовители согласно гарантийным обязательствам.
- 5.2 Гарантия изготовителя не распространяется на механические повреждения системы и не является видом работ, входящих в гарантийное обслуживание.
- 5.3 По истечению гарантийного срока, ремонт Системы и её составных частей выполняет предприятие, эксплуатирующее установку.

6. МЕРЫ БЕЗОПАСНОСТИ ПРИ ЭКСПЛУАТАЦИИ И РЕМОНТЕ

6.1 К эксплуатации и ремонту Системы допускаются лица, изучившие настоящее руководство, Правила технической эксплуатации электроустановок потребителей (ПТЭ), Правила техники безопасности при эксплуатации электроустановок потребителей (ПТБ), прошедшие инструктаж по охране труда и технике безопасности на рабочем месте. Эксплуатирующий Систему персонал должен иметь I квалификационную группу по электробезопасности, а ремонтный персонал — не ниже III.

7. ХРАНЕНИЕ СИСТЕМЫ

Подп. и дата

Инв. № дубл.

ž

Взам. инв.

Подп. и дата

Инв. № подл.

- 7.1 Для хранения Системы необходимо выполнить демонтаж и поместить в транспортную тару и упаковку предприятия-изготовителя.
- 7.2 Систему до введения в эксплуатацию следует хранить на складе в упаковке предприятия-изготовителя по группе I (Л) ГОСТ 15150-69 при температуре окружающего воздуха от плюс 5 до плюс 40 °C и относительной влажности воздуха до 80 % при температуре 25 °C.

8. ТРАНСПОРТИРОВАНИЕ

- 8.1 Система должна транспортироваться в закрытых транспортных средствах любого вида при внешних воздействиях, не превышающих следующих норм:
- температура окружающего воздуха от минус 10 до плюс 35 °C;
- относительная влажность при температуре окружающего воздуха плюс $30^{\circ}\mathrm{C}$ (65±15) %.
- 8.2 Расстановка и крепление транспортной тары с упакованной установкой в транспортных средствах должны обеспечивать её устойчивое положение и отсутствие перемещения во время транспортирования.

					ī
					ì
					ĺ
					ı
Изм.	Лист	№ докум.	Подп.	Дата	

_{Лист} 30

8.3 При транспортировании должна	быть обеспечена	защита трансп	юртной та-
ры с упакованной установкой от поп	іадания влаги.		

8.4 Остальные меры предосторожности следует осуществлять согласно сигнальной маркировке на таре.

9. УТИЛИЗАЦИЯ УСТАНОВКИ

9.1 Утилизация установки производится в порядке, принятом на предприятии, эксплуатирующем установку. Веществ, способных нанести вред здоровью человека или окружающей среде, в составе установки не содержится.

Подп. и дата								
Инв. № дубл.								
Взам. инв. №								
Подп. и дата								
Инв. № подл.			Лист	№ докум.	Подп.	Дата		Лист
	k	Копиро	овал				Формат /	A4

```
10. ИСХОДНЫЙ КОД ПРОГРАММЫ-УТИЛИТЫ ПУЛЬТА ПНУ
 #include <ansi c.h>
 #include "toolbox.h"
 #include <udpsupp.h>
 #include <cvirte.h>
 #include <userint.h>
 #include "opu.h"
 static int ph;
 #define
          STR SIZE
                                   1000
 #define
          UDP_TIMEOUT
                                   5000 // Максимальное время ожидания ответа в мс
 #define
          UDP_CLEAN_TIMEOUT 200
                                         // Таймаут очистки входного "мусора", если мы игнорируем па-
 кеты
             addr str[100] = "192.168.6.2"; // Строка с текущим IP адресом
 char
 unsigned int UDP_chan = 0;
                                         // Идентификатор открытого UDP соединения
 unsigned int UDP_port = 0;
                                         // Порт UDP соединения
 int pcnt = 1; // Счетчик отправляемых пакетов
 // Подключить/отключить UDP соединение
 int connect_UDP_GUI ( int link ) {
    int st;
    unsigned char addr [4];
    // Сначала отключаем UDP, даже если он был подключен
    if (UDP_chan) { DisposeUDPChannel (UDP_chan); UDP_chan = 0; }
    if (link) {
       // Теперь подключаем, если есть такое требование
       // читаем адрес из полей окна:
       GetCtrlVal (ph, PANEL_IPA0, &addr[0]);
       GetCtrlVal (ph, PANEL IPA1, &addr[1]);
       GetCtrlVal (ph, PANEL_IPA2, &addr[2]);
       GetCtrlVal (ph, PANEL_IPA3, &addr[3]);
       // Создаем адресную строку
       sprintf (addr_str, "%i.%i.%i.%i", addr[0], addr[1], addr[2], addr[3]);
       // читаем порт из полей окна
       GetCtrlVal (ph, PANEL_UDP_PORT, &UDP_port);
       // пытаемся открыть порт
       st = CreateUDPChannel (UDP_port, &UDP_chan);
       if (st) {
          MessagePopup ("", "Ошибка создания UDP соединения!");
          UDP_chan = 0;
          return -1;
    return 0;
 }
          DUMP_SIZE 65536
 unsigned char dump[DUMP_SIZE]; // Для вычитывания проигнорированных датаграмм UDP
 // Почистить входной буфер UDP от проигнорированных ответов
 int clean_udp_input ( unsigned long timeout ) {
    int size;
    if (!UDP_chan) {
       MessagePopup ("", "Нет UDP соединения!");
                                                                                                         Лист
Изм. Лист
            № докум.
                         Подп.
                                  Дата
```

№ дубл.

Инв.

윋

Взам. инв.

Подп. и дата

№ подл.

```
return -1;
    ProcessSystemEvents();
    while (1) {
       DisableBreakOnLibraryErrors ();
       size = UDPRead (UDP_chan, 0, 0, timeout, NULL, NULL); // Вычитываем размер датаграммы
       EnableBreakOnLibraryErrors ();
       if (size <= 0) break; // Нет датаграммы - выходим из цикла
       // Вычитываем саму датаграмму
       if (size >= DUMP_SIZE) {
          UDPRead (UDP_chan, dump, DUMP_SIZE, UDP_TIMEOUT, NULL, NULL);
       } else {
          UDPRead (UDP_chan, dump, size, 0, NULL, NULL);
    return 0;
 }
 ///////// 1. ПРОЧИТАТЬ СТАТУС /////////
 typedef struct {
    unsigned short
                    Command;
                                  // код команды 0х0100
    unsigned short
                    nPacket;
                                  // порядковый номер пакета
   tReadStatus;
 typedef struct {
   unsigned short
                      Command:
                                     // код ответа 0х8100
   unsigned short
                      nPacket;
                                // порядковый номер пакета
   unsigned short
                      Error;
                                  // поле ошибок блока перемещений
   unsigned short
                      nPointBuf;
                                     // количество точек перемещения в буфере
   unsigned short
                      MoveUnitState;
                                        // текущее состояние блока перемещений
   unsigned short
                      StatErrorA;
                                        // копия состояния оси азимута
   unsigned short
                      CoordinateA;
   short
                      SpeedA;
   unsigned short
                      StatErrorE; // копия состояния оси наклона
   unsigned short
                      CoordinateE:
   short
                      SpeedE;
                      VccPwrAE;
   unsigned short
 } tReplyStatus;
 int read_status_1 (void) {
    int st, size;
    char str[STR_SIZE];
    tReadStatus trd; // Отправляемая структура для запроса статуса
    tReplyStatus rrd; // Принимаемая структура для запроса статуса
    // Чистим экран ответа
    ResetTextBox (ph, PANEL_TEXTBOX, "");
    // Чистим входные UDP сообщения, которые мы могли проигнорировать
    if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
    // Запрос статуса
    trd.Command = 0x0100;
    trd.nPacket = pcnt;
    рспt++; // увеличиваем счетчик пакетов на 1
    st = UDPWrite(UDP chan, UDP port, addr str, &trd, sizeof(trd));
                                                                                                        Лист
Изм. Лист
            № докум.
                         Подп.
                                 Дата
```

№ дубл.

Инв.

윋

Взам. инв.

Подп. и дата

Инв. № подл.

```
if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP chan = 0; return -1; }
    // Ожидание ответа
    size = UDPRead (UDP chan, &rrd, sizeof(rrd), UDP TIMEOUT, NULL, NULL);
    if (size != sizeof(rrd) || rrd.Command != 0x8100) {
       return -2; // Ошибка приема - не бъется размер или содержимое
    sprintf (str, "N пакета = %i", rrd.nPacket); InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, str);
    sprintf (str, "Поле ошибок = %i", rrd.Error); InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, str);
    sprintf (str, "Cocт. блока перемещений = %i",
                                                  rrd.MoveUnitState);
    InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, str);
    sprintf (str, "Количество точек в буфере = %i\n", rrd.nPointBuf);
    InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, str);
    sprintf (str, "Cocт. ocu азимута = %i",
                                               rrd.StatErrorA);
    InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, str);
    sprintf (str, "Координата азимута = %i",
                                               rrd.CoordinateA);
    InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, str);
    sprintf (str, "Сост. оси наклона = %i",
                                               rrd.StatErrorE);
    InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, str);
    sprintf (str, "Координата оси наклона = %i",
                                                  rrd.CoordinateE);
    InsertTextBoxLine (ph, PANEL TEXTBOX, -1, str);
    return 0; // Успешное выполнение
 }
//////// 2. ДВИГАТЬСЯ В ТОЧКУ ////////
typedef struct {
   unsigned short
                   Command:
                                      // код команды 0x0101/ответ 0x8101
   unsigned short
                   nPacket;
                                   // порядковый номер пакета
                                   // координата для поворота в горизонтальной плоскости
   unsigned short
                   Azimuth;
   unsigned short
                   Elevator;
                                   // координата для поворота в вертикальной плоскости
 } tMoveToPoint;
 // Вход:
 //
       interactive = 0 - обычный код для перемещения в точку, с ожиданием ответа от ОПУ по UDP
 //
       interactive = 1 - укороченный код для перемещения в точку, без ожиданий ответа от ОПУ по UDP с
 целью быстрого асинхронного управления
 int move_to_point_2 (int interactive) {
    int st, size;
    tMoveToPoint trd; // Отправляемая структура для перемещения в точку
    tMoveToPoint rrd; // Принимаемая структура
    if (!interactive) {
       // Чистим экран ответа
       ResetTextBox (ph, PANEL_TEXTBOX, "");
       // Чистим входные UDP сообщения, которые мы могли проигнорировать
       if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
    } else {
       // Чистим входные UDP сообщения, которые мы могли проигнорировать без ожидания
       if (clean_udp_input(0)) return -1;
    // Запрос статуса
    trd.Command = 0x0101;
    trd.nPacket = pcnt;
    рспt++; // увеличиваем счетчик пакетов на 1
    // Вычитываем положение наклона и горихонтального углов из GUI
                                                                                                          Лист
Изм. Лист
            № докум.
                          Подп.
                                  Дата
```

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп. и

№ подл.

Инв.

```
GetCtrlVal ( ph, PANEL_AZIMUTH_ANGLE, &trd.Azimuth );
    GetCtrlVal ( ph, PANEL_ELEVATION_ANGLE, &trd.Elevator );
    st = UDPWrite (UDP chan, UDP port, addr str, &trd, sizeof(trd));
    if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP_chan = 0; return -1; }
    if (!interactive) {
       size = UDPRead (UDP chan, &rrd, sizeof(rrd), UDP TIMEOUT, NULL, NULL);
       if (size < 0 \parallel size != sizeof(rrd) \parallel rrd.Command != 0x8101) {
          MessagePopup ("","Ошибочный ответ!\nВозможно:\n1.ОПУ не инициализировано.\nПараметры
 команды вне допустимых пределов.\n");
          return -2; // Ошибка приема - не бьется размер или содержимое
    }
    return 0; // Успешное окончание
 }
 typedef struct {
   unsigned short
                  Command;
                                    // код команды = 0x0102
   unsigned short
                  nPacket;
                                    // порядковый номер пакета
   unsigned short
                  nPointMoveLoad;
                                    // порядковый номер загружаемой точки
   unsigned short
                                    // координата для поворота в горизонтальной плоскости
                  Azimuth;
   unsigned short
                  Elevator;
                                    // координата для поворота в вертикальной плоскости
 } tLoadPointBuf;
 typedef struct _UDPReplyLoadPointBuf {
   unsigned short
                  Command:
                                       // код команды = 0x8102
   unsigned short
                  nPacket;
                                    // порядковый номер пакета
                  nPointMoveLoad; // число точек в буфере
   unsigned short
 } tReplyLoadPointBuf;
 // Команда загрузки точки последовательности
 int load_point_3 ( void ) {
    char str[STR_SIZE];
    int st, size;
    unsigned short n;
    tLoadPointBuf trd;
    tReplyLoadPointBuf rrd;
    // Чистим экран ответа
    ResetTextBox (ph, PANEL_TEXTBOX, "");
    // Чистим входные UDP сообщения, которые мы могли проигнорировать
    if (clean udp input(UDP CLEAN TIMEOUT)) return -1;
    // Запрос статуса
    trd.Command = 0x0102;
    trd.nPacket = pcnt;
    pcnt++; // увеличиваем счетчик пакетов на 1
    // Вычитываем положение наклона и горихонтального углов из GUI
    GetCtrlVal (ph, PANEL_AZIMUTH_ANGLE, &trd.Azimuth);
    GetCtrlVal (ph, PANEL ELEVATION ANGLE, &trd.Elevator);
    GetCtrlVal (ph, PANEL L NPOINT, &n);
    trd.nPointMoveLoad = n:
    SetCtrlVal (ph, PANEL L NPOINT, n+1);
                                                                                                     Лист
Изм. Лист
           № докум.
                        Подп.
                                Дата
```

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп. и

№ подл.

```
st = UDPWrite (UDP chan, UDP port, addr str, &trd, sizeof(trd));
   if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP_chan = 0; return -1; }
   size = UDPRead (UDP chan, &rrd, sizeof(rrd), UDP TIMEOUT, NULL, NULL);
   if (size < 0 \parallel size != sizeof(rrd) \parallel rrd.Command != 0x8102) {
      MessagePopup ("","Ошибочный ответ!\nВозможно:\nОПУ не инициализировано\nбуфер уже содержит
 точки последовательности\пнеправильная нумерация пакетов\п"
         "Выполните останов, чтобы инициализировать массив подгружаемых точек!");
      return -2; // Ошибка приема - не бъется размер или содержимое
   sprintf (str, "Hoмep UDP пакета = %i\nЧисло точек в буфере = %i", rrd.nPacket, rrd.nPointMoveLoad );
 InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, str);
   return 0; // Успешное окончание
 }
 //////// 4. ЗАПУСК ПОСЛЕДОВАТЕЛЬНОСТИ ////////////
 typedef struct {
   unsigned short
                  Command; // код команды 0x0110, ответ 0x8110
   unsigned short
                  nPacket;
                             // порядковый номер пакета
 } tStart;
 // Запустить проигрывание буфера
int start movement 4 (void) {
   int st, size;
   tStart trd:
   tStart rrd;
   // Чистим экран ответа
   ResetTextBox (ph, PANEL_TEXTBOX, "");
   // Чистим входные UDP сообщения, которые мы могли проигнорировать
   if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
   // Запрос статуса
   trd.Command = 0x0110;
   trd.nPacket = pcnt;
   pcnt++; // увеличиваем счетчик пакетов на 1
   st = UDPWrite (UDP_chan, UDP_port, addr_str, &trd, sizeof(trd));
   if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP_chan = 0; return -1; }
   size = UDPRead (UDP_chan, &rrd, sizeof(rrd), UDP_TIMEOUT, NULL, NULL);
   if (size < 0 \parallel size != sizeof(rrd) \parallel rrd.Command != 0x8110) {
      MessagePopup ("","Ошибочный ответ! Возможно, ОПУ не инициализировано или иные ошибки!");
      return -2; // Ошибка приема - не бьется размер или содержимое
   return 0; // Успешное окончание
 }
 typedef struct {
                                   // код команды 0x0111, код ответа 0x8111
   unsigned short
                  Command:
   unsigned short
                  nPacket:
                                // порядковый номер пакета
                                                                                                  Лист
Изм. Лист
           № докум.
                        Подп.
                               Дата
```

дата

Подп. и

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп. и

№ подл.

```
} tStop;
        // Остановить проигрывание буфера
        int stop_movement_5 ( void ) {
           int st, size;
           tStop trd;
           tStop rrd;
           // Чистим экран ответа
           ResetTextBox (ph, PANEL_TEXTBOX, "");
           // Чистим входные UDP сообщения, которые мы могли проигнорировать
           if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
           // Запрос статуса
           trd.Command = 0x0111;
           trd.nPacket = pcnt;
           рспt++; // увеличиваем счетчик пакетов на 1
           st = UDPWrite (UDP_chan, UDP_port, addr_str, &trd, sizeof(trd));
           if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP_chan = 0; return -1; }
           size = UDPRead (UDP_chan, &rrd, sizeof(rrd), UDP_TIMEOUT, NULL, NULL);
           if (size < 0 \parallel size != sizeof(rrd) \parallel rrd.Command != 0x8111) {
              MessagePopup ("","Ошибочный ответ! Возможно, ОПУ не инициализировано или иные ошибки!");
              return -2; // Ошибка приема - не бъется размер или содержимое
           return 0; // Успех
        }
        /* UDP пакет, команда установки параметров рабочего ускорения и ограничения скорости привода */
Подп. и дата
        typedef struct {
                                               // код команды: 0x0140 / ответ: 0x8140
           unsigned short
                          Command;
           unsigned short
                          nPacket;
                                            // порядковый номер пакета
           unsigned short
                          MaxAccelAzimuth;
                                               // максимальное ускорение по азимуту (в горизонтальной плос-
        кости)
           unsigned short
                         MaxAccelElevator;
                                              // максимальное ускорение по наклону (в вертикальной плоско-
        сти)
№ дубл.
                         MaxVelocityAzimuth; // максимальная скорость по азимуту (в горизонтальной плоско-
           unsigned short
        сти)
           unsigned short
                         MaxVelocityElevator; // максимальная скорость по наклону (в вертикальной плоско-
Инв.
        сти)
        } tSetAccVel;
윋
        int set_accvel_6 (void) {
Взам. инв.
           int st, size;
           char str[STR_SIZE];
           tSetAccVel trd; // Отправляемая структура для запроса статуса
           tSetAccVel rrd; // Принимаемая структура для запроса статуса
           // Чистим экран ответа
дата
           ResetTextBox (ph, PANEL_TEXTBOX, "");
Подп. и
           // Чистим входные UDP сообщения, которые мы могли проигнорировать
           if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
           // Запрос статуса
№ подл.
                                                                                                            Лист
       Изм. Лист
                   № докум.
                                Подп.
                                        Дата
```

```
trd.Command = 0x0140;
   trd.nPacket = pcnt;
   рспt++; // увеличиваем счетчик пакетов на 1
   GetCtrlVal (ph, PANEL AZIMUTH AMAX, &trd.MaxAccelAzimuth);
   GetCtrlVal (ph, PANEL_ELEVATION_AMAX, &trd.MaxAccelElevator);
   GetCtrlVal (ph, PANEL AZIMUTH VMAX, &trd.MaxVelocityAzimuth);
   GetCtrlVal (ph, PANEL_ELEVATION_VMAX, &trd.MaxVelocityElevator);
   st = UDPWrite(UDP chan, UDP port, addr str, &trd, sizeof(trd));
   if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP_chan = 0; return -1; }
   // Ожидание ответа
   size = UDPRead (UDP_chan, &rrd, sizeof(rrd), UDP_TIMEOUT, NULL, NULL);
   if (size != sizeof(rrd) || rrd.Command != 0x8140) {
      MessagePopup ("", "Ошибочный ответ!");
      return -2; // Ошибка приема - не бъется размер или содержимое
   }
   // Прописываем принятые значения назад в GUI
   SetCtrlVal (ph, PANEL_AZIMUTH_AMAX, rrd.MaxAccelAzimuth );
   SetCtrlVal (ph, PANEL_ELEVATION_AMAX, rrd.MaxAccelElevator );
   SetCtrlVal (ph, PANEL_AZIMUTH_VMAX, rrd.MaxVelocityAzimuth);
   SetCtrlVal (ph, PANEL_ELEVATION_VMAX, rrd.MaxVelocityElevator);
   return 0; // Успешное выполнение
 }
 // Структура для запроса параметров на чтение
 typedef struct {
   unsigned short
                  Command;
                                   // код команды 0х0140
   unsigned short
                  nPacket;
                                // порядковый номер пакета
 } tReadAccVel;
 int read_accvel_7 (void) {
   int st, size;
   char str[STR SIZE];
   tReadAccVel trd:
   tSetAccVel rrd:
   // Чистим экран ответа
   ResetTextBox (ph, PANEL_TEXTBOX, "");
   // Чистим входные UDP сообщения, которые мы могли проигнорировать
   if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
   // Запрос статуса
   trd.Command = 0x0140;
   trd.nPacket = pcnt;
   pcnt++; // увеличиваем счетчик пакетов на 1
   st = UDPWrite(UDP_chan, UDP_port, addr_str, &trd, sizeof(trd));
   if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP_chan = 0; return -1; }
   // Ожидание ответа
   size = UDPRead (UDP chan, &rrd, sizeof(rrd), UDP TIMEOUT, NULL, NULL);
   if ( size != sizeof(rrd) || rrd.Command != 0x8140 ) {
      MessagePopup ("", "Ошибочный ответ!");
      return -2; // Ошибка приема - не бъется размер или содержимое
                                                                                                  Лист
Изм. Лист
           № докум.
                        Подп.
                               Дата
```

Подп. и

№ дубл.

Инв.

윋

ИНВ.

Взам.

дата

Z

Подп.

№ подл.

```
}
   // Прописываем принятые значения назад в GUI
   SetCtrlVal (ph, PANEL_AZIMUTH_AMAX, rrd.MaxAccelAzimuth );
   SetCtrlVal (ph, PANEL_ELEVATION_AMAX, rrd.MaxAccelElevator );
   SetCtrlVal (ph, PANEL_AZIMUTH_VMAX, rrd.MaxVelocityAzimuth );
   SetCtrlVal (ph, PANEL ELEVATION VMAX, rrd.MaxVelocityElevator);
   return 0; // Успешное выполнение
 }
 // UDP пакет, команда установки параметров смещений и ограничений по углу для привода
 typedef struct {
   unsigned short
                  Command;
                                // код команды 0х0120
   unsigned short
                  nPacket;
                             // порядковый номер пакета
   signed short
                  OffsetAzimuth; // смещение в горизонтальной плоскости
   signed short
                  OffsetElevator; // смещение в вертикальной плоскости
   unsigned short
                  MinElevator; // минимальная координата в вертикальной плоскости
   unsigned short
                  MaxElevator; // максимальная координата в вертикальной плоскости
 } tSetOffsetLimit;
 // Структура для ответа параметров смещение и ограничений по углу для привода
 typedef struct {
   unsigned short
                                   // код ответа 0х8120
                     Reply;
   unsigned short
                     nPacket;
                                   // порядковый номер пакета
   short
                     OffsetAzimuth; // смещение 0 в горизонтальной плоскости
   short
                     OffsetElevator; // смещение 0 в вертикальной плоскости
   unsigned short
                                   // минимальная координата в вертикальной плоскости
                     MinElevator;
   unsigned short
                     MaxElevator:
                                   // максимальное значение координаты в вертикальной плоскости
 } tReplyOffsetLimit;
 int write_offset_8 (void) {
   int st, size;
   char str[STR_SIZE];
   tSetOffsetLimit trd;
   tReplyOffsetLimit rrd;
   // Чистим экран ответа
   ResetTextBox (ph, PANEL_TEXTBOX, "");
   // Чистим входные UDP сообщения, которые мы могли проигнорировать
   if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
   // Запрос статуса
   trd.Command = 0x0120;
   trd.nPacket = pcnt;
   pcnt++; // увеличиваем счетчик пакетов на 1
   GetCtrlVal (ph, PANEL_OFFSET_AZIMUTH, &trd.OffsetAzimuth);
                                                                     // смещение 0 в горизонталь-
 ной плоскости
   GetCtrlVal (ph, PANEL_OFFSET_ELEVATION, &trd.OffsetElevator);
                                                                  // смещение 0 в вертикальной
 плоскости
   GetCtrlVal (ph, PANEL_ELEVATION_MIN, &trd.MinElevator);
                                                                  // минимальная координата в
 вертикальной плоскости
    GetCtrlVal (ph, PANEL ELEVATION MAX, &trd.MaxElevator);
                                                                     // максимальное значение
 координаты в вертикальной плоскости
   st = UDPWrite(UDP chan, UDP port, addr str, &trd, sizeof(trd));
                                                                                                  Лист
           № докум.
Изм. Лист
                        Подп.
                               Дата
```

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп. и ;

№ подл.

```
if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP chan = 0; return -1; }
    // Ожидание ответа
    size = UDPRead (UDP chan, &rrd, sizeof(rrd), UDP TIMEOUT, NULL, NULL);
    if ( size != sizeof(rrd) || rrd.Reply != 0x8120 ) {
       MessagePopup ("", "Ошибочный ответ!");
       return -2; // Ошибка приема - не бъется размер или содержимое
    }
    // Прописываем принятые значения назад в GUI
                                                                     // смещение 0 в горизонтальной
    SetCtrlVal (ph, PANEL_OFFSET_AZIMUTH, rrd.OffsetAzimuth );
    SetCtrlVal (ph, PANEL_OFFSET_ELEVATION, rrd.OffsetElevator );
                                                                     // смещение 0 в вертикальной
 плоскости
    SetCtrlVal (ph, PANEL_ELEVATION_MIN, rrd.MinElevator);
                                                                     // минимальная координата в
 вертикальной плоскости
    SetCtrlVal (ph, PANEL_ELEVATION_MAX, rrd.MaxElevator);
                                                                     // максимальное значение коор-
 динаты в вертикальной плоскости
    return 0; // Успешное выполнение
 }
 ///////// 9. ПРОЧИТАТЬ СМЕЩЕНИЯ И ОГРАНИЧЕНИЯ УГЛОВ ПОВОРОТА //////////
// UDP пакет, команда чтения параметров смещений и ограничений по углу для привода
 typedef struct {
   unsigned short Command;
                               // код команды 0х0120
                           // порядковый номер пакета
   unsigned short nPacket;
 } tReadOffsetLimit;
 int read_offset_9 ( void ) {
    int st, size;
    char str[STR_SIZE];
    tReadOffsetLimittrd;
    tReplyOffsetLimit rrd;
    // Чистим экран ответа
    ResetTextBox (ph, PANEL_TEXTBOX, "");
    // Чистим входные UDP сообщения, которые мы могли проигнорировать
    if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
    // Запрос статуса
    trd.Command = 0x0120;
    trd.nPacket = pcnt;
    рспt++; // увеличиваем счетчик пакетов на 1
    st = UDPWrite(UDP chan, UDP port, addr str, &trd, sizeof(trd));
    if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP_chan = 0; return -1; }
    // Ожидание ответа
    size = UDPRead (UDP_chan, &rrd, sizeof(rrd), UDP_TIMEOUT, NULL, NULL);
    if ( size != sizeof(rrd) || rrd.Reply != 0x8120 ) {
       MessagePopup ("", "Ошибочный ответ!");
       return -2; // Ошибка приема - не бъется размер или содержимое
    // Прописываем принятые значения назад в GUI
    SetCtrlVal (ph, PANEL OFFSET AZIMUTH, rrd.OffsetAzimuth );
                                                                     // смещение 0 в горизонтальной
 плоскости
                                                                                                      Лист
Изм. Лист
            № докум.
                         Подп.
                                Дата
```

Подп. и д

№ дубл.

Инв.

윋

ИНВ.

Взам.

дата

Подп. и

№ подл.

```
SetCtrlVal (ph, PANEL OFFSET ELEVATION, rrd.OffsetElevator);
                                                                // смещение 0 в вертикальной
 плоскости
   SetCtrlVal (ph, PANEL_ELEVATION_MIN, rrd.MinElevator);
                                                                 // минимальная координата в
 вертикальной плоскости
   SetCtrlVal (ph, PANEL_ELEVATION_MAX, rrd.MaxElevator);
                                                                 // максимальное значение коор-
 динаты в вертикальной плоскости
   return 0; // Успешное выполнение
 }
 typedef struct {
   unsigned short
                  Command; // код команды: 0x01F0 / ответ: 0x81F0
   unsigned short
                  nPacket; // порядковый номер пакета
   tReset;
int make reset 10 (void) {
   int st, size;
   char str[STR_SIZE];
   tReset trd;
   tReset rrd;
   // Чистим экран ответа
   ResetTextBox (ph, PANEL_TEXTBOX, "");
   // Чистим входные UDP сообщения, которые мы могли проигнорировать
   if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
   // Запрос статуса
   trd.Command = 0x01F0;
   trd.nPacket = pcnt;
   pcnt++; // увеличиваем счетчик пакетов на 1
   st = UDPWrite(UDP_chan, UDP_port, addr_str, &trd, sizeof(trd));
   if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP_chan = 0; return -1; }
   SetWaitCursor (1):
   // Ожидание ответа
   size = UDPRead (UDP_chan, &rrd, sizeof(rrd), 40000.0, NULL, NULL);
   SetWaitCursor (0);
   if (size != sizeof(rrd) || rrd.Command != 0x81F0) {
      return -2; // Ошибка приема - не бъется размер или содержимое
   // Выводим ответ на экран
   sprintf (str, "N пакета = %i", rrd.nPacket); InsertTextBoxLine (ph, PANEL TEXTBOX, -1, str);
   return 0; // Успешное выполнение
 }
 typedef struct {
   unsigned short
                               // код команды = 0x0F00
                 Command:
   unsigned short
                 nPacket;
                            // порядковый номер пакета
                                                                                               Лист
Изм. Лист
           № докум.
                       Подп.
                              Дата
```

Подп. и 🛭

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп. и

№ подл.

```
unsigned char IpAdr[4];
                             // новый IP адрес
    unsigned char CopyIpAdr[4]; // копия нового IP адреса
 } tSetIP;
 typedef struct {
    unsigned short
                                // код ответа 0x8F00
                    Reply;
    unsigned short
                    nPacket;
                                // порядковый номер пакета
    tReplySetIP;
 int set_UDP_IP_11 (void) {
    int st, size;
    char str[STR_SIZE];
    tSetIP trd;
    tReplySetIP rrd;
    // Чистим экран ответа
    ResetTextBox (ph, PANEL_TEXTBOX, "");
    // Чистим входные UDP сообщения, которые мы могли проигнорировать
    if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
    st = ConfirmPopup ("", "Эта операция поменяет IP адрес устройства,\n"
       "Обязательно запомните и запишите будущее значение IP адреса\n"
       "Если не уверены, то лучше этого не делать!\n"
       "Нажмите Yes - для выполнения или No - для отмены\n"
       );
    if (st) {
       // Запрос статуса
       trd.Command = 0x0F00;
       trd.nPacket = pcnt;
       pcnt++; // увеличиваем счетчик пакетов на 1
       GetCtrlVal (ph, PANEL_NIPA0, &trd.IpAdr[0]); GetCtrlVal (ph, PANEL_NIPA0, &trd.CopyIpAdr[0]);
       GetCtrlVal (ph, PANEL_NIPA1, &trd.IpAdr[1]); GetCtrlVal (ph, PANEL_NIPA1, &trd.CopyIpAdr[1]);
       GetCtrlVal (ph, PANEL_NIPA2, &trd.IpAdr[2]); GetCtrlVal (ph, PANEL_NIPA2, &trd.CopyIpAdr[2]);
       GetCtrlVal (ph, PANEL_NIPA3, &trd.IpAdr[3]); GetCtrlVal (ph, PANEL_NIPA3, &trd.CopyIpAdr[3]);
       st = UDPWrite(UDP chan, UDP port, addr str, &trd, sizeof(trd));
       if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP_chan = 0; return -1; }
       // Ожидание ответа
       size = UDPRead (UDP_chan, &rrd, sizeof(rrd), UDP_TIMEOUT, NULL, NULL);
       if (size != sizeof(rrd) || rrd.Reply != 0x8F00) {
          MessagePopup ("","Ошибка изменения номера адреса!");
          return -2; // Ошибка приема - не бъется размер или содержимое
       }
       // Вроде как все успешно
       connect UDP GUI (0);
       InsertTextBoxLine (ph,
                                  PANEL_TEXTBOX,
                                                              "Новое
                                                                        зачение
                                                                                  успешно
                                                                                              послано
 ОПУ.\nПодключение UDP закрыто\n");
       SetCtrlVal (ph, PANEL_IPA0, trd.IpAdr[0]); SetCtrlVal (ph, PANEL_IPA1, trd.IpAdr[1]);
       SetCtrlVal (ph, PANEL_IPA2, trd.IpAdr[2]); SetCtrlVal (ph, PANEL_IPA3, trd.IpAdr[3]);
       if (!connect_UDP_GUI (1)) {
          InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, "Смена IP адреса успешна.\nНовое подключение
 UDP открыто\n");
    return 0: // Успешное выполнение
                                                                                                             Лист
Изм. Лист
            № докум.
                          Подп.
                                   Дата
```

Подп. и

№ дубл.

Инв.

инв. №

Взам. 1

дата

Z

Подп.

№ подл.

```
}
 //////// 12. ПОМЕНЯТЬ UDP АДРЕС ///////////
 typedef struct {
   unsigned short
                  Command;
                                  // код команды 0x01FF
   unsigned short
                  nPacket;
                               // порядковый номер пакета
   unsigned short
                  UdpPort;
                               // новый UDP порт
   unsigned short
                  CopyUdpPort; // копия нового UDP порта
 } tSetUDPPort;
 typedef struct {
    unsigned short
                               // код ответа 0x81FF
                   Reply;
    unsigned short
                  nPacket;
                               // порядковый номер пакета
   tReplySetUDPPort;
int set UDP port 12 (void) {
    int st, size;
    char str[STR_SIZE];
    tSetUDPPort
                  trd; // Отправляемая структура для запроса статуса
    tReplySetUDPPort rrd; // Принимаемая структура для запроса статуса
    // Чистим экран ответа
    ResetTextBox (ph, PANEL_TEXTBOX, "");
    // Чистим входные UDP сообщения, которые мы могли проигнорировать
    if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
    st = ConfirmPopup ("", "Эта операция поменяет UDP порт,\n"
       "Обязательно запомните и запишите будущее значение UDP порта\n"
       "Если не уверены, то лучше этого не делать!"
       "Нажмите Yes - для выполнения или No - для отмены\n"
       );
    if (st) {
       // Запрос статуса
       trd.Command = 0x01FF;
       trd.nPacket = pcnt;
       рспt++; // увеличиваем счетчик пакетов на 1
       GetCtrlVal (ph, PANEL_NEW_UDP_PORT, &trd.UdpPort);
       GetCtrlVal (ph, PANEL_NEW_UDP_PORT, &trd.CopyUdpPort);
       st = UDPWrite(UDP_chan, UDP_port, addr_str, &trd, sizeof(trd));
       if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP_chan = 0; return -1; }
       // Ожидание ответа
       size = UDPRead (UDP_chan, &rrd, sizeof(rrd), UDP_TIMEOUT, NULL, NULL);
       if (size != sizeof(rrd) || rrd.Reply != 0x81FF) {
          MessagePopup ("","Ошибка изменения номера порта!");
          return -2; // Ошибка приема - не бьется размер или содержимое
       // Вроде как все успешно
       connect_UDP_GUI (0);
                                PANEL TEXTBOX,
       InsertTextBoxLine
                         (ph,
                                                          "Новое
                                                                   зачение
                                                                             успешно
                                                                                        послано
 ОПУ.\пПодключение UDP закрыто\n");
       SetCtrlVal (ph, PANEL_UDP_PORT, trd.UdpPort);
       if (!connect_UDP_GUI (1)) {
                                                                                                      Лист
Изм. Лист
                         Подп.
                                 Дата
```

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп. и

Инв. № подл.

№ докум.

```
InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, "Смена порта успешна.\nНовое подключение UDP
 открыто\n");
    return 0; // Успешное выполнение
 }
 int main (int argc, char *argv[]) {
    int st;
    // Запуск графического интерфейса
    if (InitCVIRTE (0, argv, 0) == 0)
       return -1; /* out of memory */
    if ((ph = LoadPanel (0, "opu.uir", PANEL)) < 0)
       return -1;
    DisplayPanel (ph);
    RunUserInterface (); // Запуск обработчика оконных событий
    DiscardPanel (ph);
    return 0;
 }
 //////// Функции обработки нажимаемых кнопок GUI /////////
 // Кнопка Выход
 int CVICALLBACK quit (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT_COMMIT:
           QuitUserInterface (0);
          break;
    return 0;
 // Кнопка чтения статуса
 int CVICALLBACK status_request (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT_COMMIT:
           read_status_1 ();
          break;
    return 0;
 }
 // Кнопка сброса
 int CVICALLBACK reset_request (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT_COMMIT:
           make_reset_10();
          break;
    return 0;
 }
                                                                                                             Лист
Изм. Лист
             № докум.
                           Подп.
                                   Дата
```

№ дубл.

Инв.

윋

Взам. инв.

Подп. и дата

Инв. № подл.

```
// Кнопка смены UDP порта
 int CVICALLBACK setUDP_PORT_request (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT_COMMIT:
           set_UDP_port_12();
          break;
    }
    return 0;
 }
 // Кнопка смены ІР адреса
 int CVICALLBACK setUDP_IP_request (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT_COMMIT:
           set_UDP_IP_11 ();
          break;
    return 0;
 }
 // Кнопка создания UDP соединения
 int CVICALLBACK connect (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT_COMMIT:
          // Создать UDP соединение
          // st = CreateUDPChannelConfig (UDP_PORT, "", 1, UDPCallback, NULL, &UDP_chan);
          connect_UDP_GUI (1);
          break;
    return 0;
 }
 // Кнопка закрытия UDP соединения
 int CVICALLBACK disconnect (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT COMMIT:
           connect_UDP_GUI (0);
           break;
    return 0;
 }
 // Кнопка установки смещений и границ
 int CVICALLBACK set_offset_request (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    if (event == EVENT_COMMIT) {
       write_offset_8 ();
    return 0;
 }
 // Кнопка чтения смещений и границ
 int CVICALLBACK read_offset_request (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    if (event == EVENT_COMMIT) {
       read offset 9();
                                                                                                                Лист
Изм. Лист
             № докум.
                           Подп.
                                    Дата
```

№ дубл.

Инв.

윋

Взам. инв.

Подп. и дата

№ подл.

```
return 0:
 }
 // Кнопка "двигаться в точку"
 int CVICALLBACK move_to_point_request (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT_COMMIT:
           move_to_point_2(0);
           break;
    return 0;
 }
 // Кнопка "загрузить точку"
 int CVICALLBACK load_point_request (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT COMMIT:
           load_point_3 ();
           break;
    return 0;
 }
 // Кнопка "запустить последовательность"
 int CVICALLBACK start_movement_request (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT_COMMIT:
           start_movement_4 ();
           break;
    return 0;
 // Кнопка "остановить последовательность"
 int CVICALLBACK stop_movement_request (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT_COMMIT:
           stop_movement_5();
           break:
    return 0;
 }
 // Кнопка "двигаться в точку", если включен интерактивный режим, по слайдерам
 int CVICALLBACK follow angle (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    int interactive;
    switch (event) {
       case EVENT_VAL_CHANGED:
           GetCtrlVal (ph, PANEL_INTERACTIVE, &interactive);
           if (interactive) {
              move_to_point_2(1);
           break;
    return 0:
                                                                                                                 Лист
Изм. Лист
             № докум.
                           Подп.
                                    Дата
```

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп. и

№ подл.

```
}
        // Кнопка "установки максимальных скоростей и ускорений"
        int CVICALLBACK set_accvel_request (int panel, int control, int event,
              void *callbackData, int eventData1, int eventData2) {
           switch (event) {
              case EVENT COMMIT:
                set accvel 6();
                break;
           }
           return 0;
        }
        // Кнопка "чтения максимальных скоростей и ускорений"
        int CVICALLBACK read_accvel_request (int panel, int control, int event,
              void *callbackData, int eventData1, int eventData2) {
           switch (event) {
              case EVENT_COMMIT:
                read accvel 7();
                break;
           return 0;
        }
        ///////// Функции и переменные для управления экспериментом потоковой загрузки ОПУ через UDP
        int experiment\_busy = 0;
        int work = 0;
                                        // Флаг для останова эксперимента кнопкой останов
                                        // 1 - Эксперимент ведется через подгрузку последовательности, точки
        int exp_sync_mode = 0;
Подп. и дата
        которой "проигрываются" на ОПУ каждые 10мс
                                        // 0 - Эксперимент ведется асинхронно, РС по собственному таймеру
        посылает на ОПУ асинхронные координаты и та их отрабатывает
                                             // Граничное значение подгружаемого угла азимута, в сотнях
        unsigned short exp azimuth 1;
        мкРад (единицах угла), при режиме качания
        unsigned short exp_azimuth_2;
                                              // Граничное значение подгружаемого угла азимута, в сотнях
№ дубл.
        мкРад (единицах угла), при режиме качания
                                           // Граничное значение подгружаемого угла наклона, в сотнях мкРад
        unsigned short
                      exp_elevation_1;
Инв.
        unsigned short
                      exp_elevation_2;
                                           // Граничное значение подгружаемого угла наклона, в сотнях мкРад
        int
                exp_azimuth_rotation = 0; // 1 - Азимут вращается, знак вращения определяется знаком скорости
윋
                                        // 0 - Азимут качается, границы качания определяются минимальным и
Взам. инв.
        максимальным значениями
        double
                   exp_azimuth_rate = 0.0;
                                          // Величина скорости изменения угла азимута [единицы угла/сек]
        double
                   exp_elevation_rate = 0.0; // Величина скорости изменения угла наклона [единицы угла/сек]
                                           // Величина расчетного значения азимута (с плавающей точкой), в
        double
                   exp_azimuth_calc;
дата
        единицах угла
                                             // Текущие округленные (целочисленные) значения азимута, в
        unsigned short exp_azimuth_calc_us;
Подп. и ,
        единицах угла, посылаемые в ОПУ
                azimuth dir = 0;
                                     // Направление текущее качания азимута
№ подл.
                                                                                                          Лист
Инв.
       Изм. Лист
                  № докум.
                               Подп.
                                       Дата
```

```
double
            exp_elevation_calc;
                                     // Величина расчетного значения наклона (с плавающей точкой), в
единицах угла
unsigned short exp_elevation_calc_us;
                                        // Текущие округленные (целочисленные) значения наклона, в
единицах угла, посылаемые в ОПУ
         elevation_dir = 0;
                                  // Направление текущее качания наклона
int
#define
               TIME QUANT 0.01
                                           // Период между отсчетами координаты
double
            new_time;
                                  // Текущее время в секундах, для привязки отправляемых данных к
времени РС
double
            thr_time;
                                  // Временной порог в секундах, для привязки отправляемых данных к
времени РС
         exp_init = 0;
                               // Флаг что эксперимент был проинициализован правильно
int
/// Функция для периодического вычитывания параметров оконного интерфейса пользователя
/// при выполнении потоковой загрузки
void read GUI (void) {
   // Обработать возможное изменение параметров
   ProcessSystemEvents ();
   // Вычитать интересующие параметры
   GetCtrlVal ( ph, PANEL_AZIMUTH_START, &exp_azimuth_1 );
   GetCtrlVal ( ph, PANEL_AZIMUTH_STOP, &exp_azimuth_2 );
   // Сортируем так, чтобы было удобно работать, т.е. exp azimuth 1 \le \exp azimuth 2
   if (exp_azimuth_1 > exp_azimuth_2) {
      SwapBlock (&exp azimuth 1, &exp azimuth 2, sizeof(exp azimuth 1)); // Меняем значения местами
   }
   GetCtrlVal ( ph, PANEL_ELEVATION_START, &exp_elevation_1 );
   GetCtrlVal ( ph, PANEL_ELEVATION_STOP, &exp_elevation_2 );
   // Сортируем так, чтобы было удобно работать, т.е. \exp azimuth 1 \le \exp azimuth 2
   if (exp_elevation_1 > exp_elevation_2) {
      SwapBlock (&exp_elevation_1, &exp_elevation_2, sizeof(exp_elevation_1)); // Меняем значения ме-
стами
   }
   GetCtrlVal (ph, PANEL_AZIMUTH_MODE, &exp_azimuth_rotation);
   GetCtrlVal ( ph, PANEL_AZIMUTH_RATE, &exp_azimuth_rate );
   GetCtrlVal ( ph, PANEL_ELEVATION_RATE, &exp_elevation_rate );
   // Отобразить текущие расчетные координаты осей
   SetCtrlVal (ph, PANEL_AZIMUTH_CALC, exp_azimuth_calc_us);
   SetCtrlVal (ph, PANEL_ELEVATION_CALC, exp_elevation_calc_us);
}
// Пересчитать читать новые углы (положения) за один временной квант (10мс)
// эти углы будут подгружаться асинхронно или синхронно в ПНУ
void calc_new_angles ( void ) {
   // Расчет наклона - обычное качание
      if (!elevation_dir) {
         exp_elevation_calc += exp_elevation_rate*TIME_QUANT;
      } else {
         exp_elevation_calc -= exp_elevation_rate*TIME_QUANT;
      if (exp_elevation_calc > exp_elevation_2) { // Отражение от большей границы
         exp elevation calc = exp elevation 2;
                                             // Ограничение координаты
         elevation dir = !elevation dir;
                                           // Меняем направление движения на противоположное
                                                                                                        Лист
```

Подп. и

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп.

№ подл.

Изм. Лист

№ докум.

Подп.

Дата

```
}
               if (exp_elevation_calc < exp_elevation_1) { // Отражение от меньшей границы
                  exp_elevation_calc = exp_elevation_1;
                                                         // Ограничение координаты
                                                      // Меняем направление движения на противоположное
                  elevation dir = !elevation dir;
               exp elevation calc us = exp elevation calc; // Округляем координату для дальнейшей отправки в ось
            // Расчет азимута
            if (exp_azimuth_rotation) { // Включено вращение
               exp_azimuth_calc += exp_azimuth_rate*TIME_QUANT;
               if (exp_azimuth_calc < 0) exp_azimuth_calc += 2*PI*10000;
                                                                                // Добавляем период
               if (exp_azimuth_calc >= 2*PI*10000) exp_azimuth_calc -= 2*PI*10000; // Вычитаем период
            } else { // Обычное качание
               if (!azimuth_dir) {
                   exp_azimuth_calc += exp_azimuth_rate*TIME_QUANT;
                   exp azimuth calc -= exp azimuth rate*TIME QUANT;
               if (exp_azimuth_calc > exp_azimuth_2) { // Отражение от большей границы
                  exp_azimuth_calc = exp_azimuth_2; // Ограничение координаты
                                                   // Меняем направление движения на противоположное
                   azimuth_dir = !azimuth_dir;
               }
               if (exp_azimuth_calc < exp_azimuth_1) { // Отражение от меньшей границы
                   exp azimuth calc = exp azimuth 1; // Ограничение координаты
                   azimuth dir = !azimuth dir;
                                                   // Меняем направление движения на противоположное
            exp_azimuth_calc_us = exp_azimuth_calc; // Округляем координату для дальнейшей отправки в ось
         }
         // Сделать асинхронное перемещение без ожидания ответа с минимальной латентностью
Подп. и дата
         int move_async (void) {
            int st, size;
            tMoveToPoint trd; // Отправляемая структура для перемещения в точку
            tMoveToPoint rrd; // Принимаемая структура
            // Запрос на перемещение
            trd.Command = 0x0101;
№ дубл.
            trd.nPacket = pcnt;
            pcnt++; // увеличиваем счетчик пакетов на 1
Инв.
            trd.Azimuth = exp_azimuth_calc_us;
            trd.Elevator = exp_elevation_calc_us;
윋
            st = UDPWrite (UDP_chan, UDP_port, addr_str, &trd, sizeof(trd));
Взам. инв.
            if (st) {
               MessagePopup ("",
                   "Ошибка отправки запроса!\n"
                   "Работа эксперимента будет прекращена!"
               );
               work = 0;
дата
               return -1; // Вернуть неудачу
Z
Подп. л
            // Игнорируем UDP отклик, иначе передача UDP по в этом режиме жутко затормозится (по крайней мере
         в данной среде разработки - LabWindows)
№ подл.
                                                                                                                      Лист
        Изм. Лист
                     № докум.
                                   Подп.
                                           Дата
```

```
return 0: // Успех
 }
 int upload_sequence ( void );
 int stop_sequence ( void );
 // Функция по нажатию кнопки "запуска" тестирования UDP потока
 int CVICALLBACK start experiment (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    int size:
    switch (event) {
       case EVENT_COMMIT:
          if (!experiment_busy) {
             // Проверка подключенного UDP соединения и чистка входного мусора, если он есть
             // Чистим входные UDP сообщения, которые мы могли проигнорировать
             if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;
             // Установить флаг, что эксперимент запущен
             experiment busy = 1;
             work = 1;
             exp init = 0; // Эксперимент пока еще не проинициализован
             // Скрываем панелями графический интерфейс,
             // чтобы заблокировать допуск к их кнопкам и некоторым другим параметрам
             SetCtrlAttribute (ph, PANEL_DECOR1, ATTR_VISIBLE, 1);
             SetCtrlAttribute (ph, PANEL_UDP_TEST_MODE, ATTR_DIMMED, 1); // Ограничить доступ
 к кнопке типа эксперимента
             // Вычитываем и устанавливаем начальные значения
             GetCtrlVal (ph, PANEL_UDP_TEST_MODE, &exp_sync_mode);
             GetCtrlVal (ph, PANEL_AZIMUTH_CALC, &exp_azimuth_calc_us); exp_azimuth_calc =
 exp_azimuth_calc_us;
             GetCtrlVal (ph, PANEL_ELEVATION_CALC, &exp_elevation_calc_us); exp_elevation_calc =
 exp_elevation_calc_us;
             azimuth_dir = 0;
                                  // Направление текущее качания азимута
             elevation_dir = 0;
                                     // Направление текущее качания наклона
             read GUI();
             // Инциализируем время
             thr_time = Timer() + TIME_QUANT;
             // Главный цикл, пока work = 1
             while (work) {
                new_time = Timer ();
                if (new_time >= thr_time) {// Прошла задержка, превышающая 10мс
                   thr time += TIME QUANT; // Устанавливаем слендующий порог во времени через оче-
 редные 10мс
                   if (exp_sync_mode) {
                      // РЕЖИМ СИНХРОННОГО ЭКСПЕРИМЕНТА с ПОДГРУЗКОЙ и выполнением
 координат по ВНУТРЕННЕМУ ТАЙМЕРУ ОПУ из внутреннего буфера
                      upload_sequence (); // своевременно подгружать и выполнять последовательность
                   } else {
                      // РЕЖИМ АСИНХРОННОГО ЭКСПЕРИМЕНТА с БЕЗ ПОДГРУЗКИ и выполнение
 асинхронных координат, принятых из РС
                      calc new angles ();
                      move_async();
                                                                                                       Лист
Изм. Лист
            № докум.
                         Подп.
                                 Дата
```

№ дубл.

Инв.

윋

Взам. инв.

дата

Z

Подп.

№ подл.

Инв.

```
}
                } else {
                   // Появилось окно, программа не занята - вычитаем текущие настройки пользователя из
 GUI, чтобы скорректировать процесс
                   read_GUI();
             // Работа завершена, сворачиваемся
             if (exp_sync_mode) {
                // Режиме синхронной подгрузки надо сделать останов последовательности внутри ОПУ,
                // чтобы не ждать пока прежде загруженный буфер не доиграет до конца.
                stop_sequence ();
             }
             // Массово очистить накопленный в UDP мусор, чтобы не мешать исполнению других команд
             clean udp input (500);// 0.5 сек - ожидание пакетов, которые еще могут прилететь и мы за ни-
 ми не следили
             // Разблокируем графический интерфейс.
             SetCtrlAttribute (ph, PANEL_DECOR1, ATTR_VISIBLE, 0);
             SetCtrlAttribute (ph, PANEL_UDP_TEST_MODE, ATTR_DIMMED, 0); // Разрешить кнопку
 выбора типа эксперимента
             // Эксперимент окончен
             experiment busy = 0;
          }
          break:
    return 0;
 }
 // Кнопка останова эксперимента
 int CVICALLBACK stop_experiment (int panel, int control, int event,
       void *callbackData, int eventData1, int eventData2) {
    switch (event) {
       case EVENT COMMIT:
          work = 0; // Установить флаг в положение, при котором цикл эксперимента прекратится
          break;
    return 0;
 }
 ///////// Обработчик подгрузки и выполнения последовательности /////////////////
 #define
             EXP_NMIN 64
                               // Минимальное число точек в буфере ОПУ, меньше которого необходима
 дополнительная подгрузка
             EXP_NMAX 128
                                  // Достаточное число точек в буфере ОПУ, при котором можно прекра-
 #define
 тить подгрузку
 int Nseq;
 // Команда протокола - останов последовательности
 int stop sequence (void) {
    int st. size:
                                                                                                       Лист
Изм. Лист
            № докум.
                         Подп.
                                 Дата
```

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп. и

№ подл.

```
tStop trd;
            tStop rrd;
            trd.Command = 0x0111;
            trd.nPacket = pcnt;
            рспt++; // увеличиваем счетчик пакетов на 1
            st = UDPWrite (UDP_chan, UDP_port, addr_str, &trd, sizeof(trd));
            if (st) {
                MessagePopup ("", "Ошибка отправки запроса!");
                work = 0;
                return -1;
            }
            // Не читаем ответ, чтобы не тормозить UDP
            // size = UDPRead (UDP_chan, &rrd, sizeof(rrd), UDP_TIMEOUT, NULL, NULL);
            // if (size < 0 \parallel size != sizeof(rrd) \parallel rrd.Command != 0x8111) {
            // MessagePopup ("","Ошибочный ответ! Возможно, ОПУ не инициализировано или иные ошибки!");
            // work = 0;
            // return -2; // Ошибка приема - не бъется размер или содержимое
            //}
            return 0; // Ничего не ждем в ответ пока
         }
         // Команда протокола - запуск подгруженной последовательности
         int start_sequence ( void ) {
            int st, size;
            tStart trd;
            tStart rrd;
            trd.Command = 0x0110;
            trd.nPacket = pcnt;
            pcnt++; // увеличиваем счетчик пакетов на 1
Подп. и дата
            st = UDPWrite (UDP_chan, UDP_port, addr_str, &trd, sizeof(trd));
            if (st) { MessagePopup ("", "Ошибка отправки запроса!"); UDP_chan = 0; return -1; }
            // Не читаем ответ, чтобы не тормозить UDP
            // size = UDPRead (UDP_chan, &rrd, sizeof(rrd), UDP_TIMEOUT, NULL, NULL);
            // if (size < 0 \parallel size != sizeof(rrd) \parallel rrd.Command != 0x8110) {
№ дубл.
            // MessagePopup ("","Ошибочный ответ! Возможно, ОПУ не инициализировано или иные ошибки!");
            // return -2; // Ошибка приема - не бъется размер или содержимое
            //}
Инв.
            return 0; // Успешное окончание
         }
윋
         // Вычитать текущий размер буфера с данными, чтобы в процессе работы эксперимента решить, пора ли
Взам. инв.
         подгружать новую порцию данных
         // timeout - время в миллисекундах, сколько ждать накопленные входные пакеты, чтобы их проигнорировать
         и запустить команду
         int read_buf_size ( unsigned long clean_timeout ) {
            int st, size;
            char str[STR_SIZE];
дата
            tReadStatus trd; // Отправляемая структура для запроса статуса
            tReplyStatus rrd; // Принимаемая структура для запорса статуса
Подп. и
            // Нужна прочистка входного канала, так как мы ответы игнорируем
            clean udp input ( clean timeout );
№ подл.
                                                                                                                           Лист
        Изм. Лист
                     № докум.
                                    Подп.
                                             Дата
```

```
// Запрос статуса
   trd.Command = 0x0100;
   trd.nPacket = pcnt;
   pcnt++; // увеличиваем счетчик пакетов на 1
   st = UDPWrite(UDP_chan, UDP_port, addr_str, &trd, sizeof(trd));
      MessagePopup ("", "Ошибка отправки запроса!");
      work = 0;
      return -1;
   }
   // Ожидание ответа
   size = UDPRead (UDP_chan, &rrd, sizeof(rrd), UDP_TIMEOUT, NULL, NULL);
   if (size != sizeof(rrd) || rrd.Command != 0x8100) {
      MessagePopup ("", "Неправильный ответ на запрос о статусе!");
      work = 0;
      return -2; // Ошибка приема - не бъется размер или содержимое
   }
   return rrd.nPointBuf;// Успешное выполнение - возвращаем текущее число точек в буфере
}
// Добавить новую точку в последовательность
int put_point ( void ) {
   int st, size;
   unsigned short n;
   tLoadPointBuf trd;
                         // Отправляемая структура для перемещения в точку
   tReplyLoadPointBuf rrd;
                            // Принимаемая структура в ответ
   // Рассчитываем новое значение углов
   calc_new_angles ();
   // Запрос на помещение точки в буфер
   trd.Command = 0x0102;
   trd.nPacket = pcnt;
   рспt++; // увеличиваем счетчик пакетов на 1
   // Вычитываем положение наклона и горихонтального углов из GUI
   trd.Azimuth = exp azimuth calc us;
   trd.Elevator = exp_elevation_calc_us;
   trd.nPointMoveLoad = Nseq; // Номер точки в последовательности
   Nseq += 1;
   st = UDPWrite (UDP_chan, UDP_port, addr_str, &trd, sizeof(trd));
      MessagePopup ("", "Ошибка отправки запроса!");
      work = 0;
      return -1;
   }
   // Кладем точку, но не ждем UDP подтверждения, так как все надо делать быстро
   // size = UDPRead (UDP_chan, &rrd, sizeof(rrd), UDP_TIMEOUT, NULL, NULL);
   //if (size < 0 \parallel size != sizeof(rrd) \parallel rrd.Command != 0x8102) {
   // MessagePopup ("","Ошибочный ответ!\nВозможно:\nОПУ не инициализировано\nбуфер уже содержит
точки последовательности\пнеправильная нумерация пакетов\п"
         "Выполните останов, чтобы инициализировать массив подгружаемых точек!");
   // work = 0;
   // return -2; // Ошибка приема - не бьется размер или содержимое
   //}
                                                                                                             Лист
```

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп. и

Инв. № подл.

Изм. Лист

№ докум.

Подп.

Дата

```
return 0; // Успешное окончание
 }
 // Подгрузить последовательность точек
 int upload_sequence ( void ) {
    int k, Nbuf;
    static int counter = 0;
    if (!exp_init) {
       /// Эксперимент еще не инициализирован - подгружаем первую часть данных в пустой буфер и запус-
 каем его
       // 1. Сначала делаем останов и сброс последовательности в ПНУ
       stop_sequence ();
       Nseq = 1;
                        // Инициализируем номер отсчета
       // 2. Начинаем грузить недостающие точки
       for (k = 0; work && k < EXP_NMAX; k++) 
           // Кладем точку за точкой
           put_point();
        }
       // 3. Запускаем
       if (work) start_sequence ();
       // 4. Сбрасываем счетчик прореживания, так как нет смысла дергать ПНУ каждые 10 мс
       counter = 0;
       // Инициализация и запуск проведен
       exp_init = 1;
    } else {
       /// Обычный режим работы - подгрузка данных
       counter++; // Увеличиваем счетчик квантов времени, для прореживания доступа к стойке
       if (counter > 20) {
           // Проверка состояния стойки каждые 0.2 сек,
           // чаще не стоит, так как 100 отсчетов "тратятся" 1 секунду, а лишний раз отвлекать ПНУ - мешать
 ее работе
           counter = 0:
           // 1. Вычитываем оставшееся число точек в буфере ПНУ
           Nbuf = read_buf_size ( 100 );
 //
           printf ("%i %i\n", Nbuf, Nseq);
 //
           printf ("Buffer points = %i", Nbuf);
           if (Nbuf < 0) {
              work = 0; // Запрещаем работу и сворачиваемся
              MessagePopup ("", "Ошибка чтения количества точек в буфере!");
              return -1;
           }
           // 2. Догружаем точки до необходимого безопасного количества
           if (Nbuf < EXP_NMIN) {
              printf (" upload %i points", EXP_NMAX-Nbuf);
 //
              for (k = 0; work && k < (EXP NMAX-Nbuf); k++) {
                 // Кладем точку за точкой
                 put_point ();
              }
                                                                                                               Лист
Изм. Лист
             № докум.
                           Подп.
                                    Дата
```

№ дубл.

Инв.

윋

Взам. инв.

дата

Подп. и ,

Инв. № подл.

}
// printf ("\n"); } return 0; Лист 55 Подп. № докум. Изм. Лист Дата

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

						регистрации изме	енений			
		Номера листов (страниц) изме- заме- но- аннули-				Всего листов	№ до-	Входящий № сопроводитель-	_	-
	Изм.	нён- ных	нён-	ВЫХ	аннули- рован- ных	(страниц) в докум.	кум.	ного докум. и дата	Подп.	Дата
-										
-										
-										
-										
-										
	Изм. Ли	ict №,								