

ПОВОРОТНО-НАКЛОННОЕ УСТРОЙСТВО

Руководство по эксплуатации

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

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.

1.1 Назначение Системы и условия эксплуатации.					
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p>1.1.1 Поворотно-наклонное устройство является несущим аппаратом, обеспечивающим возможность прецизионного управления позиционированием в горизонтальной и вертикальной плоскостях.</p> <p>1.1.2 Внешнее управление поворотно-наклонным устройством, а так же информационное обеспечение осуществляется посредством команд, передаваемых по линии Ethernet протоколом UDP</p> <p>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).</p> <p>1.1.4 Условия эксплуатации поворотно-наклонного устройства представлены в Таблице 1.</p>
Изм.	Лист	№ докум.	Подп.	Дата	

Лист
4

Таблица 1. Эксплуатационные Условия

1	Окружающая среда	Предназначено для работы в лабораторном помещении
2	Окружающая эксплуатационная температура	От +10°С до +35°С
3	Температура хранения	от -40°С до + 70°С
4	Относительная влажность	93 % при 70°С
5	Минимальное атмосферное давление	эксплуатационное 800 гПа 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 сбалансированной нагрузкой	12≥ рад/с 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-	≤ 0,8А

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

	лезной нагрузке без датчиков) $U_{ном.} = 24В$	
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 $T \geq 0^{\circ}C$ и $U_{ном.} = 24В$	около 2А
5	Текущее значение потребляемого тока во время старта с максимальным полезным грузом (без датчиков) при температуре окр. среды $T \geq 0^{\circ}C$ и $U_{ном.} = 24В$	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 электропитания системы 24В	250W
3	Источник для электропитания системы	Источник: Вход: однофазн, ~ 230В, 50Гц Выход: =24В/14.6А/350W

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

Таблица 5. Прочие требования

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

1.3 Состав Системы.

1.3.1 Состав Системы приведен в таблице 6.

Таблица 6

Обозначение	Наименование	Кол.	Примечание
-------------	--------------	------	------------

Изм.	Лист	№ докум.	Подп.	Дата	Лист
					6

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

Обозначение	Наименование	Кол.	Примечание
	Поворотно-наклонное устройство (ПНУ)	2	Первоначальный IP адрес: 192.168.6.1 и 192.168.6.2 Порт: 10000
	Блок питания и связи (БПС)	2	
	Кабель питания и передачи данных	2	
	Патч-корд	3	1 шт – 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

2. УСТРОЙСТВО И РАБОТА СИСТЕМЫ.

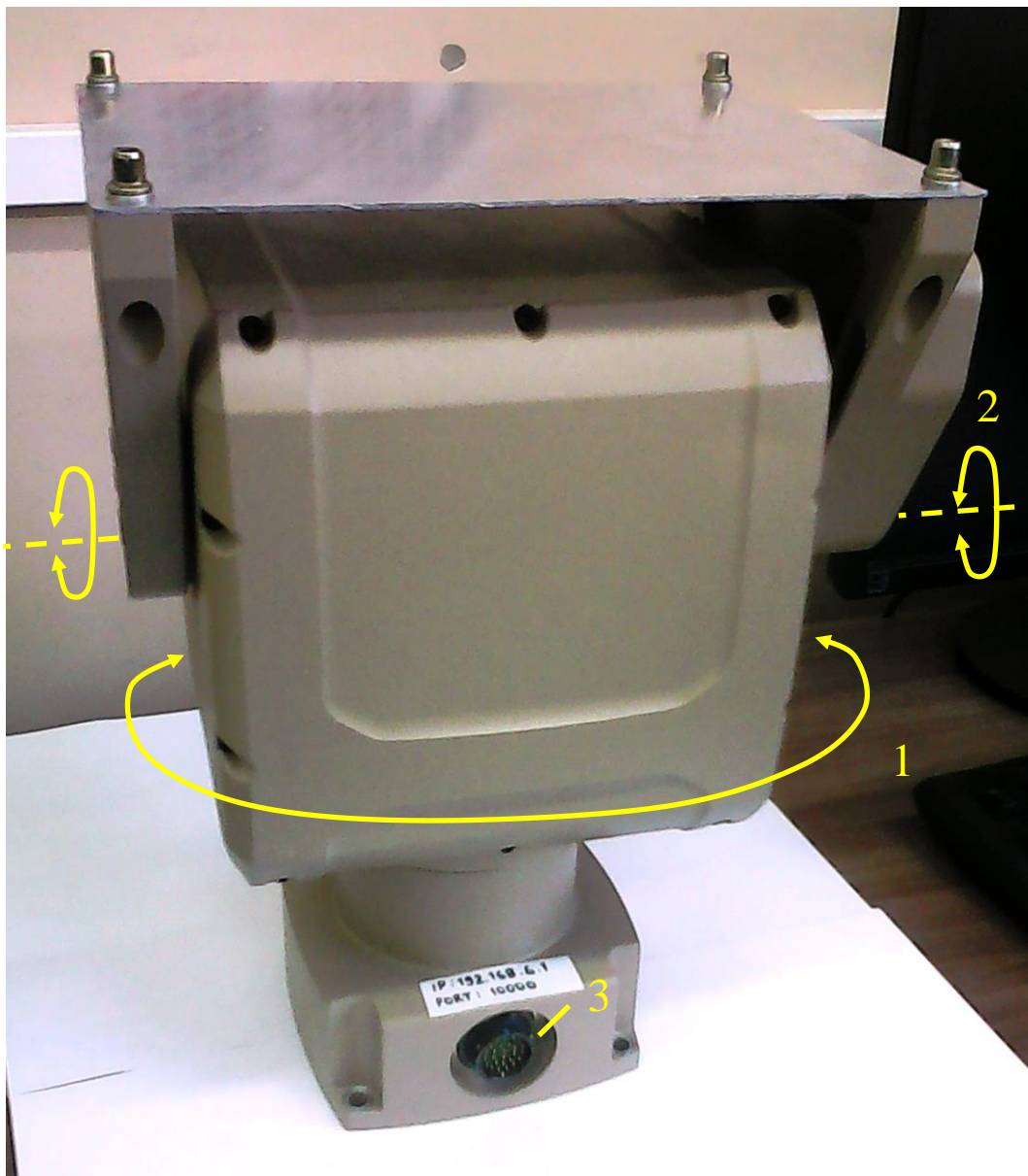
2.1 Поворотно-наклонное устройство (ПНУ).

2.1.1 Общее устройство ПНУ.

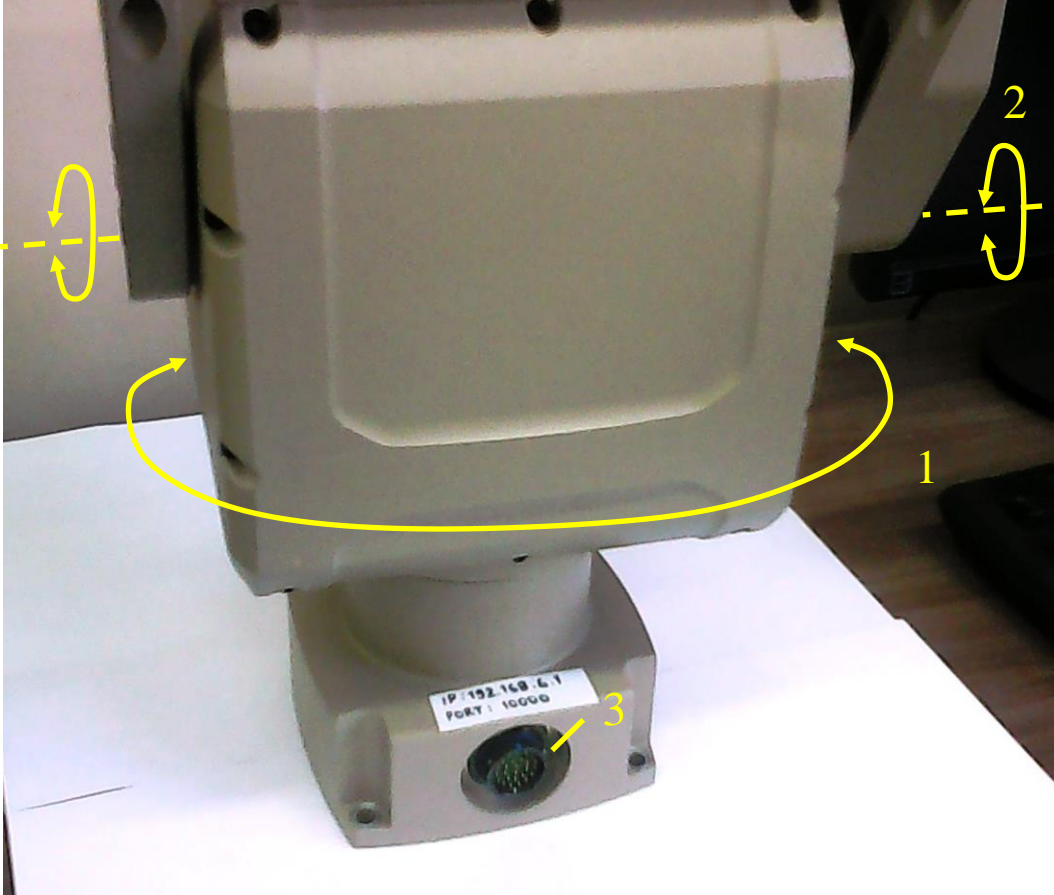
ПНУ изображено на рисунке 1. ПНУ предназначено для выполнения операций углового позиционирования тестируемого объекта и состоит из двух независимо управляемых угловых позиционеров (осей вращения):

- 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



Инв. № подл.	Подп. и дата		Взам. инв. №	Инв. № дубл.		Подп. и дата		
Изм.	Лист	№ докум.	Подп.	Дата	Лист			
					8			

					
--	--	--	--	--	--

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

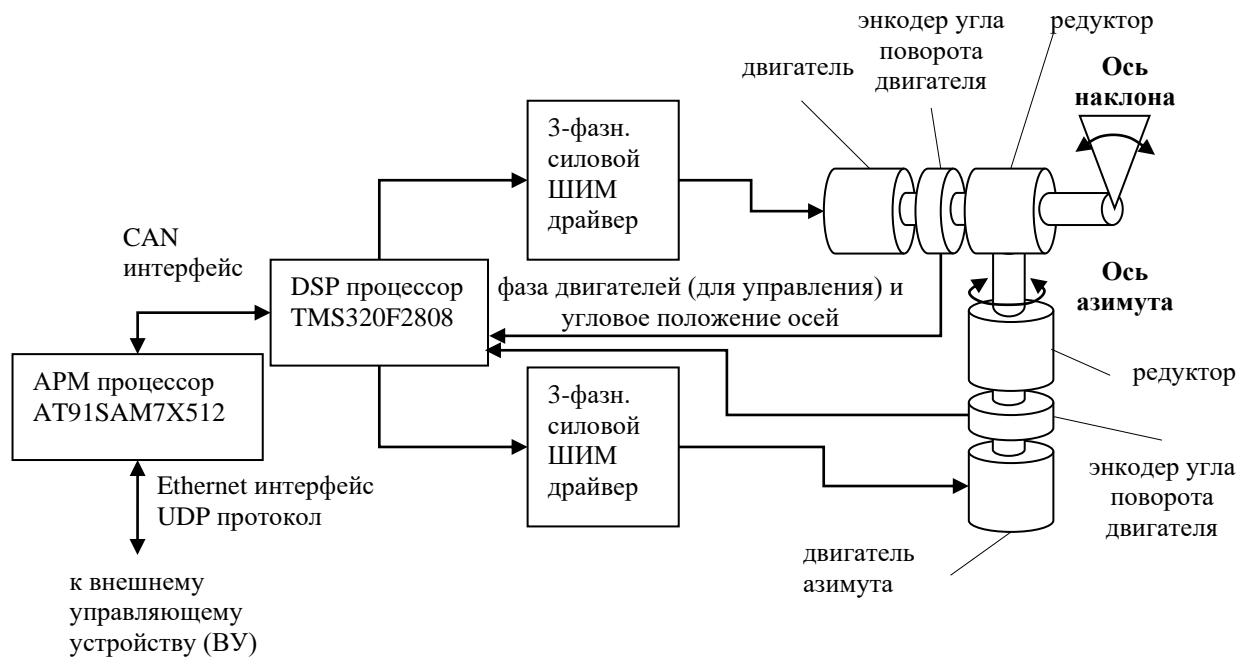


Рис. 2 Упрощенная блок-схема поворотно-наклонного устройства.

2.1.2 Управление движением ПНУ.

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

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	и синхронным:	<ul style="list-style-type: none">• 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-

Инв. № подл.	Подп. и дата				Лист 10
	Инв. № дубл.				
	Взам. инв. №				
Изм.	Подп. и дата				Формат А4
	Лист				
	№ докум.				
Копировал					

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:

- 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», соответствующие 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 электро-снабжения ПНУ (+24В, 350Вт), а также обеспечивает Ethernet соединение внешнего устройства управления с ПНУ через разъем RJ-45.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p>ирования, 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.</p> <ul style="list-style-type: none">При малом максимальном ускорении – минимизируются моменты и «удары», соответствующие 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.
<h2>2.2 Блок питания и связи (БПС).</h2>					
<p>Блок питания и связи предназначен для обеспечения силового электро-снабжения ПНУ (+24В, 350Вт), а также обеспечивает Ethernet соединение внешнего устройства управления с ПНУ через разъем RJ-45.</p>					
					Лист 11
Изм.	Лист	№ докум.	Подп.	Дата	

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



Разъем для подключения внешнего питания ~220В, 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.

2.3.1 Пультom управления является компактный PC-совместимый компьютер (мини-ноутбук), имеющий сетевую карту для Ethernet соединения и необходимое программное обеспечение для выполнения команд управления ПНУ по UDP интерфейсу. Программное обеспечение пульта будет описано ниже.

2.3.2 Пульт управления подключается к БПС через Ethernet патчкорд.

Имп. № подл.	Подп. и дата	Взам. инв. №	Инд. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

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.

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.

Инд. № подл.	Подп. и дата				Изм.	Лист	№ докум.	Подп.	Дата		Лист	
	Инд. № дубл.											13
	Взам. инв. №											
	Подп. и дата											
3.2 Подготовка Системы к использованию												
3.2.1 Распаковать Систему.												
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.												

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.

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:

№	Команда ВНУ	Краткое функциональное описание
1.	«Прочитать статус»	ПНУ: возвращает текущее состояние, для анализа пользователем системы
2.	«Двигаться в точку»	ВУ отправляет «точку» (значения азимутального и вертикального углов). ПНУ: немедленно и асинхронно (с минимальной латентностью) запускает позиционирование точку.
3.	«Загрузить точку»	ВУ: отправляет «точку» (значения азимутального и вертикального углов). ПНУ: добавляет заданные значения азимутального и вертикального углов («точку») во внутренний буфер для дальнейшего синхронного позиционирования.
4.	«Запуск последовательности»	ПНУ: Предварительно загруженная во внутренний буфер последовательность из «точек», запускается на исполнение (позиционирование) с периодичностью 10мс. По мере опустошения внутреннего буфера «точек», выполнение коман-

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

Изм.	Лист	№ докум.	Подп.	Дата	Лист
					14

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

		ды «Загрузить точку» позволяет по- полнить буфер.
5.	«Останов последовательности»	ПНУ: Выполняет останов выполнения бу- фера «точек», если он был запущен. Выполняет очистку буфера точек.
6.	«Задать максимальные ускорения и скорости»	ВУ: 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- рения и скорости»	ПНУ: Возвращает текущие значения пико- вых скоростей и рабочих ускорений для осей азимута и наклона
8.	«Задать смещения и ограничения углов поворота»	ВУ: Задаёт постоянные смещения осей при позиционировании, а также раз- решаемые пределы позиционирова- ния для оси наклона.
9.	«Прочитать смещения и ограни- чения углов поворота»	ВУ: Задаёт постоянные смещения осей 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

Изм.	Лист	№ докум.	Подп.	Дата	Лист
					15

D7 – D15 - резерв

После подачи питания ОПУ устанавливается в точку с координатой 0 по углу поворота и 31415 ($\pi \cdot 10000$) по углу наклона. Значения скорости по углу поворота и наклона представляют собой целое знаковое число. Значение младшего разряда = 100 микрорадиан/сек.

Эта команда предназначена для **асинхронных перемещений**.

ВУ посылает команду в ПНУ:

Размер структуры = 8 байт

После получения команды начинается перемещение в точку с заданным значением координат, **после достижения цели** узел управления поворотами, наклонами отправляет следующий ответ.

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 работающих сервоконтролерах (биты D0,D1 поля MoveUnitState = 0). Если значения координат выходят за пределы рабочего диапазона перемещений , принудительно устанавливаются предельные значения, бит обнаруженной ошибки в этом случае не устанавливается.

ВУ посылает в ПНУ структуру:

Размер структуры = 10 байт

Формат А4

После загрузки первой точки ОПУ перемещается на требуемую позицию и ждет окончания перемещения, все последующие значения точек при этом, загружаются в буфер, затем, по окончании 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.

typedef struct {		
unsigned short	Command;	код команды 0x8102
unsigned short	nPacket;	порядковый номер пакета
unsigned short	nPointMoveLoad;	число точек в буфере
};		

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

```
typedef struct {
    unsigned short    Command;    // код команды  0x0111
    unsigned short    nPacket;    // порядковый номер пакета
}
Размер структуры = 4 байта
```

При успешном выполнении запуска возвращается структура схожая с посылаемой:

Размер структуры = 4 байта

ВУ посылает структуру:

Размер структуры = 12 байт

Размер структуры = 12 байт

```
typedef struct {
    unsigned short    Command;           // код команды: 0x0140
    unsigned short    nPacket;          // порядковый номер пакета
}
Размер структуры = 4 байта
```

```
typedef struct {
    unsigned short    Command;           // код команды: 0x8140
    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
```

ВУ посылает структуру:

```
typedef struct {
    unsigned short    Command;      // код команды: 0x0120
    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 установки:

```
typedef struct {
    unsigned short    Command;        // код команды: 0x8120
    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
```

ВУ посылает структуру аналогичную при задании ограничений и смещения углов поворота, только урезанную:

```
typedef struct {
    unsigned short    Command;    // код команды: 0x0120
    unsigned short    nPacket;    // порядковый номер пакета
}
Размер структуры = 4 байт
```

При успешном выполнении возвращается структура, с текущими установками:

```
typedef struct {
```

						Лист
						20
Изм.	Лист	№ докум.	Подп.	Дата		

Размер структуры = 12 байт

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:

Размер структуры = 4байта

Если команда выполнена, возвращается след. ответ.

Размер структуры = 4 байта

При задании нового IP адреса, ВУ посылает структуру в ПНУ:

Размер структуры = 12 байт

```
IpAdr[3] = 2;
```

```
CopyIpAdr[3] = 2;
```

Размер структуры = 4 байт

Формат А4

* * *

```
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 0x8F00
    unsigned short    nPacket;        // порядковый номер пакета
    unsigned char     MacAdr[6];      // MAC адрес ОПУ
    unsigned char     IpAdr[4];       // IP адрес ОПУ
    unsigned short    MovePort;       // № UDP порта узла управления поворотами наклонами
    unsigned short    Com1Port;       // № UDP порта последовательного интерфейса №1
    unsigned short    Com2Port;       // № UDP порта последовательного интерфейса №2
    unsigned short    Com4Port;       // № UDP порта последовательного интерфейса №4
    unsigned short    InOutPort;      // № UDP порта узла управления подачей питания на нагрузку
};
```

Размер структуры = 24 байта

ВУ посылает структуру:

```
typedef struct {
    unsigned short Command; // код команды = 0x01FF
    unsigned short nPacket; // порядковый номер пакета
    unsigned short UdpPort; // новое значение порта
    unsigned short CopyUdpPort; // 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.

ПНУ возвращает структуру:

```
typedef struct {
    unsigned short    Command;    // код команды = 0x81FF
    unsigned short    nPacket;    // порядковый номер пакета
};
```

Размер структуры = 4 байт

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

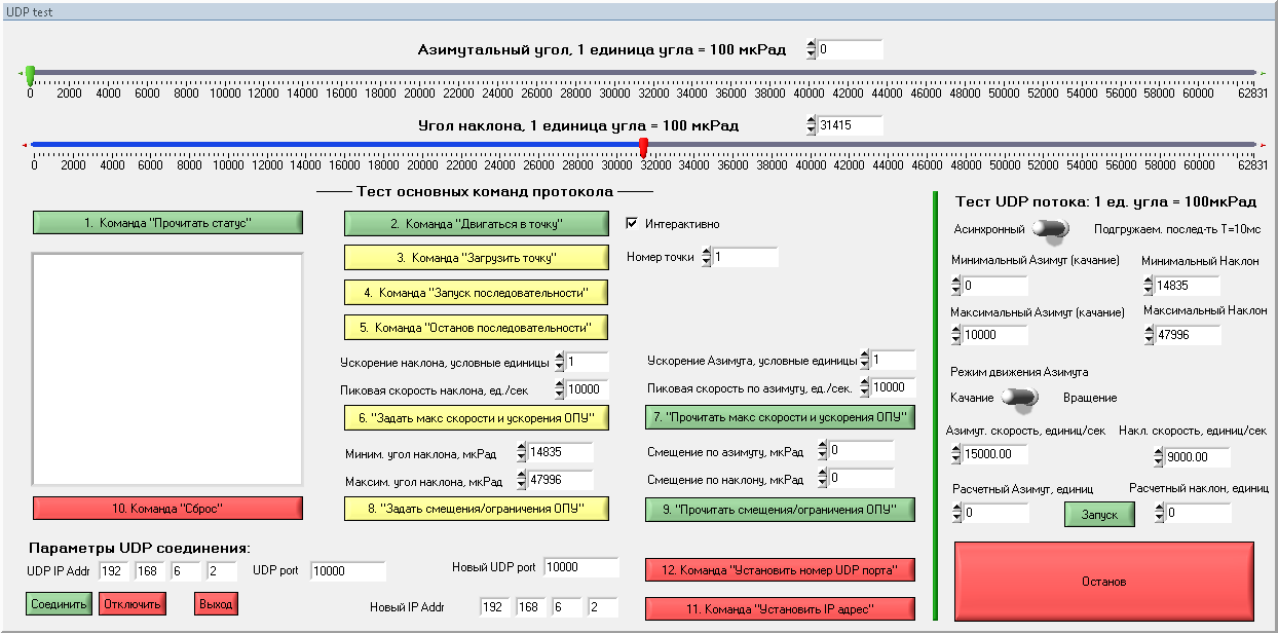
						Лист
						22
Изм.	Лист	№ докум.	Подп.	Дата		

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

Кроме того, в приложении, прилагается исходный текст утилиты на языке C, 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

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

						Лист
						23
Изм.	Лист	№ докум.	Подп.	Дата		

Тест основных команд протокола

1. Команда "Прочитать статус"

10. Команда "Сброс"

2. Команда "Двигаться в точку" ☒ Интерактивно

3. Команда "Загрузить точку" Номер точки

4. Команда "Запуск последовательности"

5. Команда "Останов последовательности"

Ускорение наклона, условные единицы Ускорение Азимута, условные единицы

Пиковая скорость наклона, ед./сек Пиковая скорость по азимуту, ед./сек

6. "Задать макс скорости и ускорения ОПУ"

7. "Прочитать макс скорости и ускорения ОПУ"

Миним. угол наклона, мкРад Смещение по азимуту, мкРад

Максим. угол наклона, мкРад Смещение по наклону, мкРад

8. "Задать смещения/ограничения ОПУ"

9. "Прочитать смещения/ограничения ОПУ"

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

Новый IP Addr 11. Команда "Установить IP адрес"

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

Тест UDP потока: 1 ед. угла = 100мкРад

Асинхронный ☒ Подгружаем. послед-ть T=10мс

Минимальный Азимут (качение)

Минимальный Наклон

Максимальный Азимут (качение)

Максимальный Наклон

Режим движения Азимута

Качение ☒ Вращение

Азимут. скорость, единиц/сек Накл. скорость, единиц/сек

Расчетный Азимут, единиц Расчетный наклон, единиц

Останов

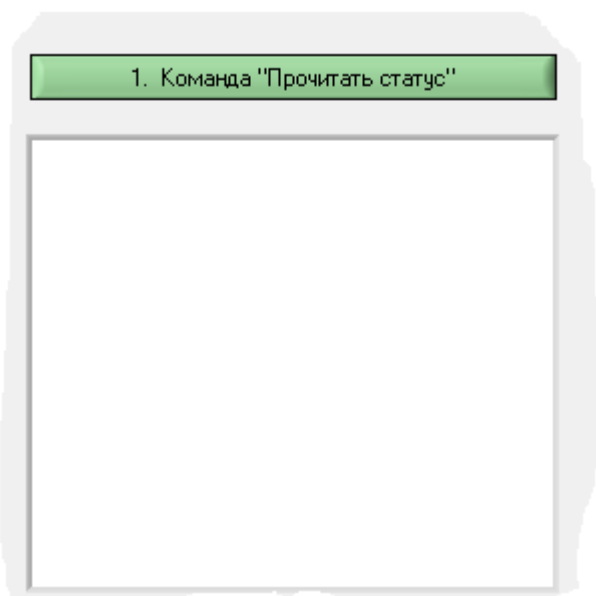
3.4.3 Краткие описания команд и элементов графического интерфейса:

3.4.3.1 Команда «1.Прочитать статус» имеет следующие элементы – кнопку и окошко, в которое выводится полученный ответ:

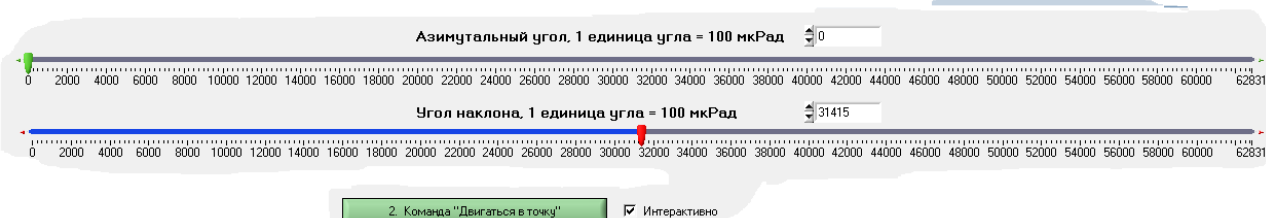
Изм.	Лист	№ докум.	Подп.	Дата

Лист

24



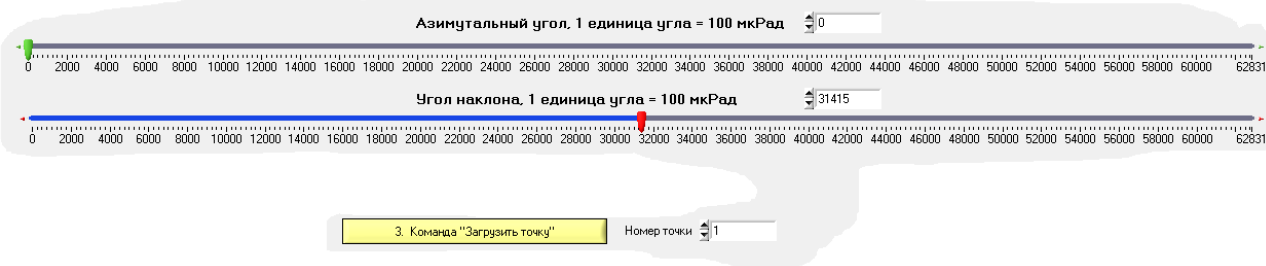
3.4.3.2 Команда «2. Двигаться в точку» имеет следующие рабочие элементы:



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 указанным в соответствующем поле.

3.4.3.4 Команда запуска последовательности расположенной в буфере ПНУ

4. Команда "Запуск последовательности"

вызывается на жатием кнопки:

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата						Лист
										25
					Изм.	Лист	№ докум.	Подп.	Дата	

3.4.3.5 Команда останова выполнения последовательности расположенной в буфере ПНУ и сброса ее вызывается нажатием кнопки:

5. Команда "Останов последовательности"

3.4.3.6 Нажатие кнопки и соответствующие поля задают пределы максимальных скоростей и ускорений осей:

Ускорение наклона, условные единицы	1	Ускорение Азимута, условные единицы	1
Пиковая скорость наклона, ед./сек	10000	Пиковая скорость по азимуту, ед./сек.	10000
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 наклона, условные единицы Ускорение Азимута, условные единицы
 Пиковая скорость наклона, ед./сек Пиковая скорость по азимуту, 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:

Миним. угол наклона, мкРад	14835	Смещение по азимуту, мкРад	0
Максим. угол наклона, мкРад	47996	Смещение по наклону, мкРад	0
8. "Задать смещения/ограничения ОПУ"			

3.4.3.9 Нажатие указанной ниже кнопки приводит к тому, что соответствующие смещения и ограничения углов читаются из ПНУ и отображаются в соответствующих значениям полях:

Миним. угол наклона, мкРад Смещение по азимуту, мкРад
 Максим. угол наклона, мкРад Смещение по наклону, мкРад

9. "Прочитать смещения/ограничения ОПУ"

3.4.3.10 Нажатие кнопки «10. Сброс» вызывает инициализацию и позиционирование осей в исходное (нулевое) положение. Все параметры осей сбрасываются в значения по умолчанию.

						Лист
						26
Изм.	Лист	№ докум.	Подп.	Дата		

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 поточковой подгрузке данных:

Асинхронный



Подгружаем. послед-ть T=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

Наклон меняется с наклонной скоростью в единицах 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.

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

						Лист
						27
Изм.	Лист	№ докум.	Подп.	Дата		

- Наклонная скорость – это скорость изменения угла наклона в единицах угла в секунду.
- Максимальное и минимальное значения – это пределы угла в которых он может изменяться.

3.4.3.15 Аналогично наклону, но независимо от него может вычисляться и задаваемый азимут:

Минимальный Азимут (качение)

Максимальный Азимут (качение)

Режим движения Азимута

☒ Качение
 ☐ Вращение

Азимут. скорость, единиц/сек

Расчетный Азимут, единиц

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



наклона.

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 При неотложной ситуации, когда доступ к выключателю затруднен – резко выдернуть шнур питания 220В из гнезда питания БПС.

Изм.	Лист	№ докум.	Подп.	Дата	
Ивн. № подл.	Подп. и дата	Взам. инв. №	Ивн. № дубл.	Подп. и дата	

Запуск

Останов

наклона.

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 При неотложной ситуации, когда доступ к выключателю затруднен – резко выдернуть шнур питания 220В из гнезда питания БПС.

Лист

28

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.

[illegible]

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

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 °С и относительной влажности воздуха до 80 % при температуре 25 °С.

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 °С;
- относительная влажность при температуре окружающего воздуха
плюс 30°С (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.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p>7.1 Для хранения Системы необходимо выполнить демонтаж и поместить в транспортную тару и упаковку предприятия-изготовителя.</p> <p>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 °С и относительной влажности воздуха до 80 % при температуре 25 °С.</p> <h3>8. ТРАНСПОРТИРОВАНИЕ</h3> <p>8.1 Система должна транспортироваться в закрытых транспортных средствах любого вида при внешних воздействиях, не превышающих следующих норм:</p> <p>- температура окружающего воздуха от минус 10 до плюс 35 °С;</p> <p>- относительная влажность при температуре окружающего воздуха плюс 30°С (65±15) %.</p> <p>8.2 Расстановка и крепление транспортной тары с упакованной установкой в транспортных средствах должны обеспечивать её устойчивое положение и отсутствие перемещения во время транспортирования.</p> <table><tr><td>Изм.</td><td>Лист</td><td>№ докум.</td><td>Подп.</td><td>Дата</td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></table>	Изм.	Лист	№ докум.	Подп.	Дата										
Изм.	Лист	№ докум.	Подп.	Дата																

8.3 При транспортировании должна быть обеспечена защита транспортной тары с упакованной установкой от попадания 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.

Инв. № подл.	Подп. и дата				Лист 31
	Инв. № дубл.				
	Взам. инв. №				
	Подп. и дата				
Изм.	Лист	№ докум.	Подп.	Дата	

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

char addr_str[100] = "192.168.6.2"; // Строка с текущим IP адресом
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) {
            MessagePopur ("", "Ошибка создания UDP соединения!");
            UDP_chan = 0;
            return -1;
        }
    }
    return 0;
}

#define 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) {
        MessagePopur ("", "Нет UDP соединения!");
    }
}
```

Подп. и дата		Изм.	Лист	№ докум.	Подп.	Дата	Лист
Изм.	Лист	№ докум.	Подп.	Дата			32

////////////////////////////////////
 ////////////////////////////////// 1. ПРОЧИТАТЬ СТАТУС ///////////////////
 //////////////////////////////////////

```
typedef struct {
    unsigned short    Command;           // код ответа 0x8100
    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;
    unsigned short    VccPwrAE;
} tReplyStatus;
```

```
// Чистим экран ответа
ResetTextBox (ph, PANEL_TEXTBOX, "");
```

```
// 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));
```

Формат А4

```

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 бьется 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, "Сост. 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, "Сост. оси 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
    unsigned short    Azimuth;           // координата для поворота в горизонтальной плоскости
    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;
    pcnt++; // увеличиваем счетчик пакетов на 1

    // Вычитываем положение наклона и горизонтального углов из GUI

```

Инва. № подл.	Подп. и дата	Инва. № дубл.	Подп. и дата	Взам. инв. №	Инва. №
---------------	--------------	---------------	--------------	--------------	---------

Изм.	Лист	№ докум.	Подп.	Дата	Лист
					34

```

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 || size != sizeof(rrd) || 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
}

////////////////////////////////////
//////////////////////////////////// 3. ЗАГРУЗИТЬ ТОЧКУ //////////////////////////////////////
////////////////////////////////////

typedef struct {
    unsigned short    Command;           // код команды = 0x0102
    unsigned short    nPacket;           // порядковый номер пакета
    unsigned short    nPointMoveLoad;    // порядковый номер загружаемой точки
    unsigned short    Azimuth;           // координата для поворота в горизонтальной плоскости
    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
    unsigned short    nPointMoveLoad;    // число точек в буфере
} 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 горизонтального 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 );

```

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

									Лист
									35
Изм.	Лист	№ докум.	Подп.	Дата					

```

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 || size != sizeof(rrd) || rrd.Command != 0x8102) {
    MessagePopup ("", "Ошибочный ответ!\nВозможно:\nОПУ не инициализировано\nбуфер уже содержит
точки последовательности\nнеправильная нумерация пакетов\n"
        "Выполните останов, чтобы инициализировать массив подгружаемых точек!");
    return -2; // Ошибка приема - не бьется размер или содержимое
}

sprintf (str, "Номер 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 || size != sizeof(rrd) || 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
}

```

```

////////////////////////////////////
//////////////////// 5. ОСТАНОВ ПОСЛЕДОВАТЕЛЬНОСТИ //////////////////////
////////////////////////////////////

```

```

typedef struct {
    unsigned short    Command;    // код команды 0x0111, код ответа 0x8111
    unsigned short    nPacket;    // порядковый номер пакета
}

```

Инов. № подл.	Подп. и дата				Лист
	Инов. № дубл.				
	Взам. инв. №				
	Подп. и дата				
Изм.	Лист	№ докум.	Подп.	Дата	<pre>// 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 size != sizeof(rrd) 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 } //////////////////////////////////// //////////////// 5. ОСТАНОВ ПОСЛЕДОВАТЕЛЬНОСТИ ////////////////// //////////////////////////////////// typedef struct { unsigned short Command; // код команды 0x0111, код ответа 0x8111 unsigned short nPacket; // порядковый номер пакета</pre>

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////// 6. ЗАДАТЬ МАКСИМАЛЬНЫЕ УСКОРЕНИЯ И СКОРОСТИ //////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/* UDP пакет, команда установки параметров рабочего ускорения и ограничения скорости привода */
typedef struct {
    unsigned short  Command;           // код команды: 0x0140 / ответ: 0x8140
    unsigned short  nPacket;           // порядковый номер пакета
    unsigned short  MaxAccelAzimuth;    // максимальное ускорение по азимуту (в горизонтальной плоско-
кости)
    unsigned short  MaxAccelElevator;   // максимальное ускорение по наклону (в вертикальной плоско-
кости)
    unsigned short  MaxVelocityAzimuth; // максимальная скорость по азимуту (в горизонтальной плоско-
кости)
    unsigned short  MaxVelocityElevator; // максимальная скорость по наклону (в вертикальной плоско-
кости)
} tSetAccVel;

int set_accvel_6 ( void ) {
    int st, size;
    char str[STR_SIZE];
    tSetAccVel trd; // Отправляемая структура для запроса статуса
    tSetAccVel rrd; // Принимаемая структура для запроса статуса

    // Чистим экран ответа
    ResefTextBox (ph, PANEL_TEXTBOX, "");

    // Чистим входные UDP сообщения, которые мы могли проигнорировать
    if (clean_udp_input(UDP_CLEAN_TIMEOUT)) return -1;

    // Запрос статуса

```

Формат А4

```

trd.Command = 0x0140;
trd.nPacket = pcnt;
pcnt++; // 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 бьется 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
}

////////////////////////////////////
//////////////// 7. ПРОЧИТАТЬ МАКСИМАЛЬНЫЕ УСКОРЕНИЯ И СКОРОСТИ //////////////////
////////////////////////////////////

// Структура для запроса параметров на чтение
typedef struct {
    unsigned short    Command;          // код команды 0x0140
    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 бьется размер или содержимое
    }
}

```

Инов. № подл.	Подп. и дата	Взам. инв. №	Инов. № дубл.	Подп. и дата						Лист
										38
					Изм.	Лист	№ докум.	Подп.	Дата	

```

}

// 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
}

////////////////////////////////////
//////////////////// 8. ЗАДАТЬ СМЕЩЕНИЯ И ОГРАНИЧЕНИЯ УГЛОВ ПОВОРОТА ///////////////////
////////////////////////////////////

// UDP пакет, команда установки параметров смещений и ограничений по углу для привода
typedef struct {
    unsigned short    Command;      // код команды 0x0120
    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    Reply;        // код ответа 0x8120
    unsigned short    nPacket;      // порядковый номер пакета
    short             OffsetAzimuth; // смещение 0 в горизонтальной плоскости
    short             OffsetElevator; // смещение 0 в вертикальной плоскости
    unsigned short    MinElevator; // минимальная координата в вертикальной плоскости
    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));
}

```

Подп. и дата	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));				
Инв. № дубл.					
Взам. инв. №					
Подп. и дата					
Инв. № подл.					
					Лист
					39
Изм.	Лист	№ докум.	Подп.	Дата	

```
// 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 бьется размер или содержимое
}
```

```
return 0; // Успешное выполнение
}
```

```
// UDP пакет, команда чтения параметров смещений и ограничений по углу для привода
typedef struct {
    unsigned short Command;      // код команды 0x0120
    unsigned short nPacket;      // порядковый номер пакета
} tReadOffsetLimit;
```

```
// Чистим экран ответа
ResetTextBox (ph, PANEL_TEXTBOX, "");
```

```
// Запрос статуса
trd.Command = 0x0120;
trd.nPacket = pcnt;
pcnt++; // увеличиваем счетчик пакетов на 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 бьется размер или содержимое
}
```

```

// Прописываем принятые значения назад в GUI
SetCtrlVal (ph, PANEL_OFFSET_AZIMUTH, rrd.OffsetAzimuth );           // смещение 0 в горизонтальной
плоскости

```



```
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
}
```

////////////////////////////////////
//////////////////////////////////// 10. СБРОС //////////////////////////////////////
////////////////////////////////////

```
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) { MessagePopur ("", "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 бьется размер или содержимое
    }

    // Выводим ответ на экран
    sprintf (str, "N пакета = %i", rrd.nPacket); InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, str);
    return 0; // Успешное выполнение
}
```

////////////////////////////////////
//////////////////////////////////// 11. ПОМЕНЯТЬ IP АДРЕС //////////////////////////////////////
////////////////////////////////////

```
typedef struct {
    unsigned short    Command; // код команды = 0x0F00
    unsigned short    nPacket; // порядковый номер пакета
}
```

Инов. № подл.	Подп. и дата				Лист
	Инов. № дубл.				
	Взам. инв. №				
Инов. № подл.	Подп. и дата				41
	Инов. № дубл.				
	Взам. инв. №				
Изм.	Лист	№ докум.	Подп.	Дата	

Подп. и дата	Подп. и дата	Инов. № дубл.	Взам. инв. №	Подп. и дата	Инов. № подл.
<pre>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 бьется размер или содержимое } // Выводим ответ на экран sprintf (str, "N пакета = %i", rrd.nPacket); InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, str); return 0; // Успешное выполнение } //////////////////////////////////// ////////// 11. ПОМЕНИТЬ IP АДРЕС ////////// //////////////////////////////////// typedef struct { unsigned short Command; // код команды = 0x0F00 unsigned short nPacket; // порядковый номер пакета }</pre>					

```

    unsigned char IpAdr[4];    // новый IP адрес
    unsigned char CopyIpAdr[4]; // копия нового IP адреса
} tSetIP;

typedef struct {
    unsigned short  Reply;    // код ответа 0x8F00
    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 бьется размер или содержимое
        }

        // Вроде как все успешно
        connect_UDP_GUI (0);
        InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, "Новое значение успешно послано в 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

```

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p>pcnt++; // увеличиваем счетчик пакетов на 1</p> <p>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]);</p> <p>st = UDPWrite(UDP_chan, UDP_port, addr_str, &trd, sizeof(trd)); if (st) { MessagePopup ("", "Ошибка отправки запроса."); UDP_chan = 0; return -1; }</p> <p>// 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 бьется размер или содержимое }</p> <p>// Вроде как все успешно connect_UDP_GUI (0); InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, "Новое значение успешно послано в 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</p>				
Изм.	Лист	№ докум.	Подп.	Дата					

Лист
42

```

////////////////////////////////////
12. ПОМЕНЯТЬ UDP АДРЕС
////////////////////////////////////

```

```
typedef struct {
    unsigned short    Reply;           // код ответа 0x81FF
    unsigned short    nPacket;         // порядковый номер пакета
} tReplySetUDPPort;
```

```
// 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 меняет 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;
    pcnt++; // 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);
InsertTextBoxLine (ph, PANEL_TEXTBOX, -1, "Новое значение успешно послано в
ОПУ.\nПодключение UDP закрыто\n");
SetCtrlVal (ph, PANEL_UDP_PORT, trd.UdpPort);
if (!connect_UDP_GUI (1)) {
```

Формат А4

```
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;
}
```

```
// 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 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;
}
```

Формат А4


```

}

// 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
/////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////// правое окно ///////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////

int experiment_busy = 0;
int work = 0; // Флаг для останова эксперимента кнопкой останов

int exp_sync_mode = 0; // 1 - Эксперимент ведется через подгрузку последовательности, точки
    // которой "проигрываются" на ОПУ каждые 10мс
    // 0 - Эксперимент ведется асинхронно, РС по собственному таймеру
    // посылает на ОПУ асинхронные координаты и та их обрабатывает

unsigned short exp_azimuth_1; // Граничное значение подгружаемого угла азимута, в сотнях
    // мкРад (единицах угла), при режиме качания
unsigned short exp_azimuth_2; // Граничное значение подгружаемого угла азимута, в сотнях
    // мкРад (единицах угла), при режиме качания

unsigned short exp_elevation_1; // Граничное значение подгружаемого угла наклона, в сотнях мкРад
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
    // 0 - Азимут качается, границы качания определяются минимальным и
    // 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]

double exp_azimuth_calc; // Величина расчетного значения азимута (с плавающей точкой), в
    // единицах угла
unsigned short exp_azimuth_calc_us; // Текущие округленные (целочисленные) значения азимута, в
    // единицах угла, посылаемые в ОПУ
int azimuth_dir = 0; // Направление текущее качания азимута

```

Подп. и дата	<pre>int work = 0; // Флаг для останова эксперимента кнопкой останов</pre> <pre>int exp_sync_mode = 0; // 1 - Эксперимент ведется через подгрузку последовательности, точки которой "проигрываются" на ОПУ каждые 10мс // 0 - Эксперимент ведется асинхронно, РС по собственному таймеру посылает на ОПУ асинхронные координаты и та их обрабатывает</pre> <pre>unsigned short exp_azimuth_1; // Граничное значение подгружаемого угла азимута, в сотнях мкРад (единицах угла), при режиме качания unsigned short exp_azimuth_2; // Граничное значение подгружаемого угла азимута, в сотнях мкРад (единицах угла), при режиме качания</pre> <pre>unsigned short exp_elevation_1; // Граничное значение подгружаемого угла наклона, в сотнях мкРад unsigned short exp_elevation_2; // Граничное значение подгружаемого угла наклона, в сотнях мкРад</pre> <pre>int exp_azimuth_rotation = 0; // 1 - Азимут вращается, знак вращения определяется знаком скорости // 0 - Азимут качается, границы качания определяются минимальным и максимальным значениями</pre> <pre>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]</pre> <pre>double exp_azimuth_calc; // Величина расчетного значения азимута (с плавающей точкой), в единицах угла unsigned short exp_azimuth_calc_us; // Текущие округленные (целочисленные) значения азимута, в единицах угла, посылаемые в ОПУ int azimuth_dir = 0; // Направление текущее качания азимута</pre>				
Инв. № дубл.					
Взам. инв. №					
Подп. и дата					
Инв. № подл.					
					<div>Лист</div> <div>47</div>
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
int         elevation_dir = 0;           // Направление текущее качания наклона

#define      TIME_QUANT 0.01             // Период между отсчетами координаты

double      new_time;                    // Текущее время в секундах, для привязки отправляемых данных к
времени PC
double      thr_time;                    // Временной порог в секундах, для привязки отправляемых данных к
времени PC

int         exp_init = 0;                 // Флаг что эксперимент был проинициализован правильно

/// 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 <= 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 <= 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
    }
}
```

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

						Лист
						48
Изм.	Лист	№ докум.	Подп.	Дата		


```

    }

    if (exp_elevation_calc < exp_elevation_1) { // Отражение от меньшей границы
        exp_elevation_calc = exp_elevation_1; // Ограничение координаты
        elevation_dir = !elevation_dir; // Меняем направление движения на противоположное
    }

    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;
    } else {
        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
        azimuth_dir = !azimuth_dir; // Меняем направление движения на противоположное
    }

    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

```

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;
trd.nPacket = pcnt;
pcnt++; // увеличиваем счетчик пакетов на 1

```

```

trd.Azimuth = exp_azimuth_calc_us;
trd.Elevator = exp_elevation_calc_us;

```

```

st = UDPWrite (UDP_chan, UDP_port, addr_str, &trd, sizeof(trd) );
if (st) {
    MessagePopup ("",
        "Ошибка отправки запроса!\n"
        "Работа эксперимента будет прекращена!"
    );
    work = 0;
    return -1; // Вернуть неудачу
}

```

// Игнорируем UDP отклик, иначе передача UDP по в этом режиме жутко затормозится (по крайней мере в данной среде разработки - LabWindows)

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

Изм.	Лист	№ докум.	Подп.	Дата	Лист
					49


```
tStop trd;
tStop rrd;
```

```
trd.Command = 0x0111;
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!");
    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 || size != sizeof(rrd) || rrd.Command != 0x8111) {
//   MessagePopur ("","Ошибочный ответ! Возможно, ОПУ не инициализировано или иные ошибки!");
//   work = 0;
//   return -2; // Ошибка приема - не бьется 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
```

```
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 || size != sizeof(rrd) || 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
```

```
// Вычитать текущий размер буфера с данными, чтобы в процессе работы эксперимента решить, пора ли
// 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];
tReadStatus trd; // Отправляемая структура для запроса статуса
tReplyStatus rrd; // Принимаемая структура для запроса статуса
```

```
// Нужна прочистка входного канала, так как мы ответы игнорируем
clean_udp_input ( clean_timeout );
```

Подп. и дата	<pre> 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; } // Не читаем ответ, чтобы не тормозить UDP // size = UDPRead (UDP_chan, &rrd, sizeof(rrd), UDP_TIMEOUT, NULL, NULL); // if (size < 0 size != sizeof(rrd) 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 } </pre>			
Инв. № дубл.				
Взам. инв. №	<pre> // Вычитать текущий размер буфера с данными, чтобы в процессе работы эксперимента решить, пора ли 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]; tReadStatus trd; // Отправляемая структура для запроса статуса tReplyStatus rrd; // Принимаемая структура для запроса статуса // Нужна прочистка входного канала, так как мы ответы игнорируем clean_udp_input (clean_timeout); </pre>			
Подп. и дата				
Инв. № подл.				
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));
if (st) {
    MessagePopur ("", "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) {
    MessagePopur ("", "Неправильный ответ на запрос о статусе!");
    work = 0;
    return -2; // Ошибка приема - не бьется 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;
    pcnt++; // увеличиваем счетчик пакетов на 1

    // Вычитываем положение наклона и горизонтального 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) );
    if (st) {
        MessagePopur ("", "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 || size != sizeof(rrd) || rrd.Command != 0x8102) {
    //    MessagePopur ("", "Ошибочный ответ!\nВозможно:\nОПУ не инициализировано\nбуфер уже содержит
точки последовательности\nнеправильная нумерация пакетов\n"
    //    "Выполните останов, чтобы инициализировать массив подгружаемых точек!");
    //    work = 0;
    //    return -2; // Ошибка приема - не бьется размер или содержимое
    //}

```

Инов. № подл.	Подп. и дата				<div>// Запрос на помещение точки в буфер trd.Command = 0x0102; trd.nPacket = pcnt; pcnt++; // 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)); if (st) { MessagePopur ("", "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 size != sizeof(rrd) rrd.Command != 0x8102) { // MessagePopur ("", "Ошибочный ответ!\nВозможно:\nОПУ не инициализировано\nбуфер уже содержит точки последовательности\nнеправильная нумерация пакетов\n" // "Выполните останов, чтобы инициализировать массив подгружаемых точек!"); // work = 0; // return -2; // Ошибка приема - не бьется размер или содержимое //}</div>
Изм.	Лист	№ докум.	Подп.	Дата	Лист
					53


```

    }
    // printf ("\n");
}

}
return 0;
}

```

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата							
Изм.	Лист	№ докум.	Подп.	Дата							

Лист
55

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