Паспорт
Контролер СКУД Castle R4Rack для IT инфраструктуры.

Контролер Castle R4Rack предназначен для построения распределенной, масштабируемой системы контроля доступа для IT инфраструктуры. Контролер позволяет управлять доступом к внутреннему объему телекоммуникационных стоек. Контролер имеет 4 зоны контроля, что позволяет контролировать или 2 рядом расположенных шкафа спереди и сзади, либо 4 рядом расположенные телекоммуникационных шкафа с одного направления.

Контролер Castle R4Rack органично вписывается в структуру системы контроля и управления доступом объектов любой сложности и масштаба. Контролер Castle R4Rack выполнен в 1U 19” корпусе, имеющем в комплекте кронштейны и крепеж для установки в стандартном телекоммуникационном шкафу. Контролер Castle R4Rack управляет специальными ручками для телекоммуникационных с встроенным управляемым запорным устройством и считывателем бесконтактных карт доступа. Контроллер контролирует положение ручки (открыто - закрыто) и положение дверей телекоммуникационного шкафа. Форматы карт которые работают с различными моделями ручек указаны в таблице 1.

Таблица 1

Castle SH-K33	MIFARE®
Castle SH-K44	Em-Marine
Castle SH-I v.2.0	MIFARE® / Em-Marine / Chekpoint

Также возможна установка управляемых ручек без считывателя, простых электромеханических замков и защелок (в зависимости от конструкции шкафа), а также любых внешних считывателей с интерфейсом Wiegand. В комплект поставки контроллера входят 4 или 6 (в зависимости от конструкции задней двери телекоммуникационного шкафа) магнитоконтактных датчика (геркона) с комплектом соединительных проводов, предназначенных для контроля положения дверей шкафа.
Контролер имеет в своем составе аккумулятор – что позволит функционировать при отсутствии сетевого питания более 2-х часов.
Для подключения периферийных устройств (замков, датчиков и т. д., а также к локальной сети в контроллере используется гнезда RJ-45. Это позволяет обеспечить простую и удобную коммутацию.

Технические характеристики Castle R4RACK:

Физические характеристики	
Габаритные размеры	415 * 215 * 48мм (без кронштейов)

Электрические характеристики		
Напряжение питания	Переменное 220 вольт ±10%.	
Потребляемый ток	Не более 0,15 А.	
Потребляемая мощность	Не более 30 Вт.	
Предельное коммутируемое напряжение силовых релейных выходов		125 В
Предельный коммутируемый ток силовых релейных выходов		12 А
Предельное коммутируемое напряжение выходов типа ОК		30 В
Предельный коммутируемый ток выходов типа ОК		0,1 А
Встроенные цепи защиты контроллера	1) Питание: a. Защита от переполюсовки питания контроллера b. Защита от перегрузки и перенапряжения цепей питания считывателей 2) Линия связи (Ethernet): a. Полная гальваническая развязка 3) Входные интерфейсы: a. Защита от переполюсовки и перенапряжения 4) Выходные интерфейсы: 5) Ограничение максимального тока и защита контактов реле от подгорания	

Интерфейсы

Линия связи	Один стандартный порт Ethernet. Скорость обмена – Ethernet 10/100BASE-TX, полный дуплекс. Поддержка протоколов DHCP, SNMP, DTLS.
Подключение считывателей	До 4 считывателей с выходным интерфейсом Wiegand или Touch memory. Поддерживаемые битности Wiegand: Wiegand-26, Wiegand-34, Wiegand-36, Wiegand-37, Wiegand-42, Wiegand-58, а также Wiegand-4, Wiegand-HID (6 бит) или Wiegand Motorola (8 бит) для подключения кодонаборных панелей
Подключение датчиков	До 10 датчиков с выходами типа «открытый коллектор» (ОК) или «сухой контакт».
Выходы индикации пульта управления	4 выхода с ОК для подключения светодиодов.
Силовые релейные выходы	4 реле, контактная группа работает на переключение

Дополнительное оборудование
Аккумулятор 12 вольт. 2,3 Ампер/часа
Комплект замков
Комплект датчиков
Комплект проводов

Условия эксплуатации	
Температура окружающего воздуха	От 0 до +45 °С
Относительная влажность воздуха	Не более 85% при t°=30°С.
Атмосферное давление	84 –106,7 кПа.
Параметры при функционировании в составе СКУД «SIGUR»	
Поддержка исполнительных устройств	Двери, оборудованные электромагнитными, электромеханическими замками или защелками.
Кол-во автономно хранимых ключей	90 000 *
Кол-во автономно хранимых событий	400 000 *
Кол-во автономно хранимых режимов доступа (временных зон)	30 000 *

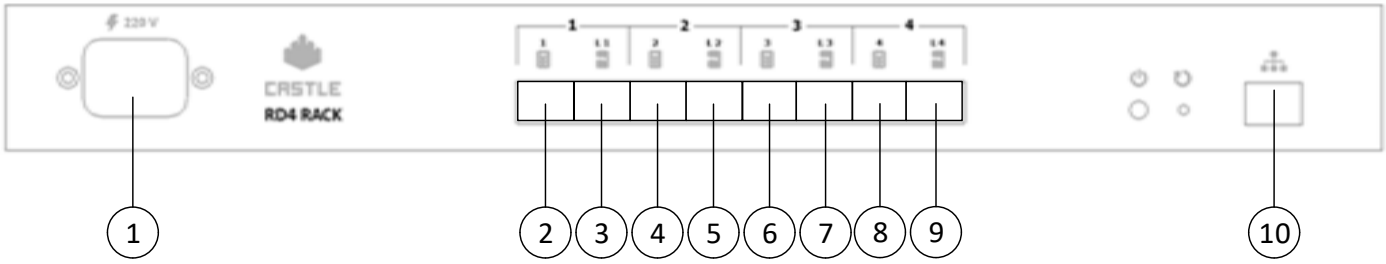
Автономная индикация состояния контроллера	1) Звуковая индикация работы контроллера и ошибок его конфигурирования 2) Визуальная индикация питания
Наличие средств обновления микропрограммы	Микропрограмма может быть обновлена через линию связи с любого компьютера, подключенного к СКУД «SIGUR». Отключение контроллера от исполнительного механизма — не требуется.

Дополнительные функции контроллера Castle R4Rack.

Весь генерируемый контроллерами Castle R4Rack протокол событий может транслироваться во внешнюю систему сбора логов в формате SYSLOG.

Контроллер Castle R4Rack поддерживает протокол SNMP, MIB файл приведен в приложении №1.

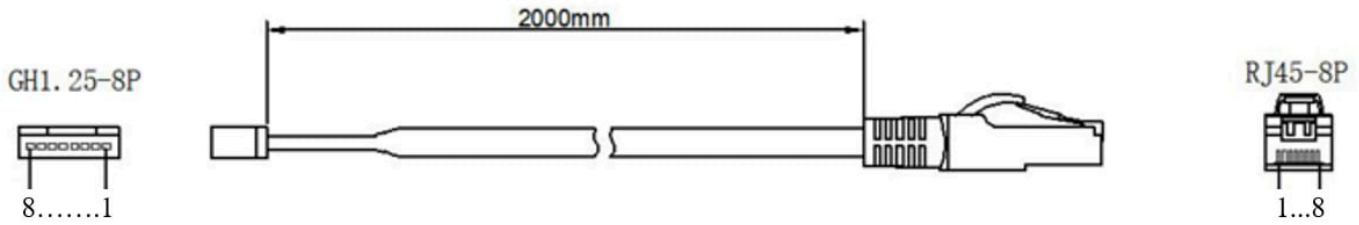
Назначение разъемов контроллера Castle R4Rack.



1. Разъем подключения питания
2. Управляемая ручка 1 дверь
3. Датчик положения 1 дверь
4. Управляемая ручка 2 дверь
5. Датчик положения 2 дверь
6. Управляемая ручка 3 дверь
7. Датчик положения 3 дверь
8. Управляемая ручка 4 дверь
9. Датчик положения 4 дверь
10. Сетевой разъем

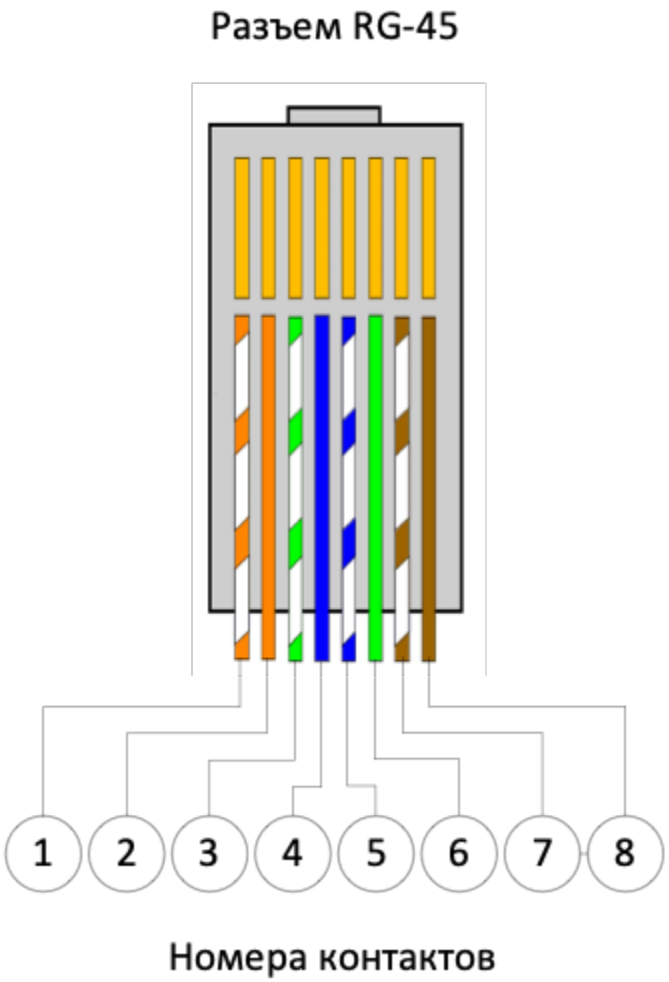
Распиновка разъемов контроллера Castle R4Rack.

Всё периферийное оборудование подключается к контроллеру Castle R2Rack при помощи стандартных разъемов RJ-45. Разделка удлиняющих проводов в разъемы выполняется по стандарту TIA/EIA-568-B. Разделка кабелей ручек имеет собственный формат.



8	Красный	+12 вольт
7	Черный	GND
6	Желтый	Wiegand D0
5	Зеленый	Wiegand D1
4	Белый	Не используется
3	Голубой	Разблокировка
2	Оранжевый	Датчик ручки
1	Коричневый	Не используется

Распиновка штатного кабеля ручки Castle SH-I v.2.0



Распиновка разъёмов подключения ручки контроллера Castle R4Rack

Разъёмы 2,4,6,8 имеют распиновку в соответствие с типом используемой с данным контроллером типа ручки – замка. Базовая распиновка выполняется для кабеля ручки Castle SH-I v.2.0:

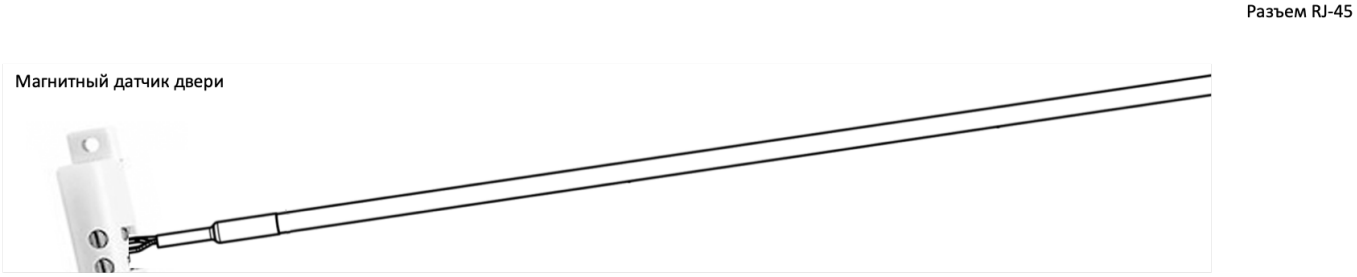
1. Не используется (Коричневый)
2. Датчик положения ручки (открытый коллектор) (Оранжевый)
3. Импульс управления замком (замок будет открыт 3 секунды после короткого импульса, или на все время длинного импульса) (Синий)
4. Не используется (Белый)
5. Wiegand D1 (Зеленый)
6. Wiegand D0 (Желтый)
7. GND (Черный)
8. +12 вольт (красный)

В скобках указаны цвета проводов кабеля входящего в комплект поставки ручки

Разъемы 3,5,7,9 для подключения магнитного датчика открытия двери имеют следующую распиновку:

1. Не используется.
2. Не используется.
3. Не используется.
4. Контакт 1 геркона
5. Не используется.
6. Не используется.
7. Не используется.
8. Контакт 2 геркона

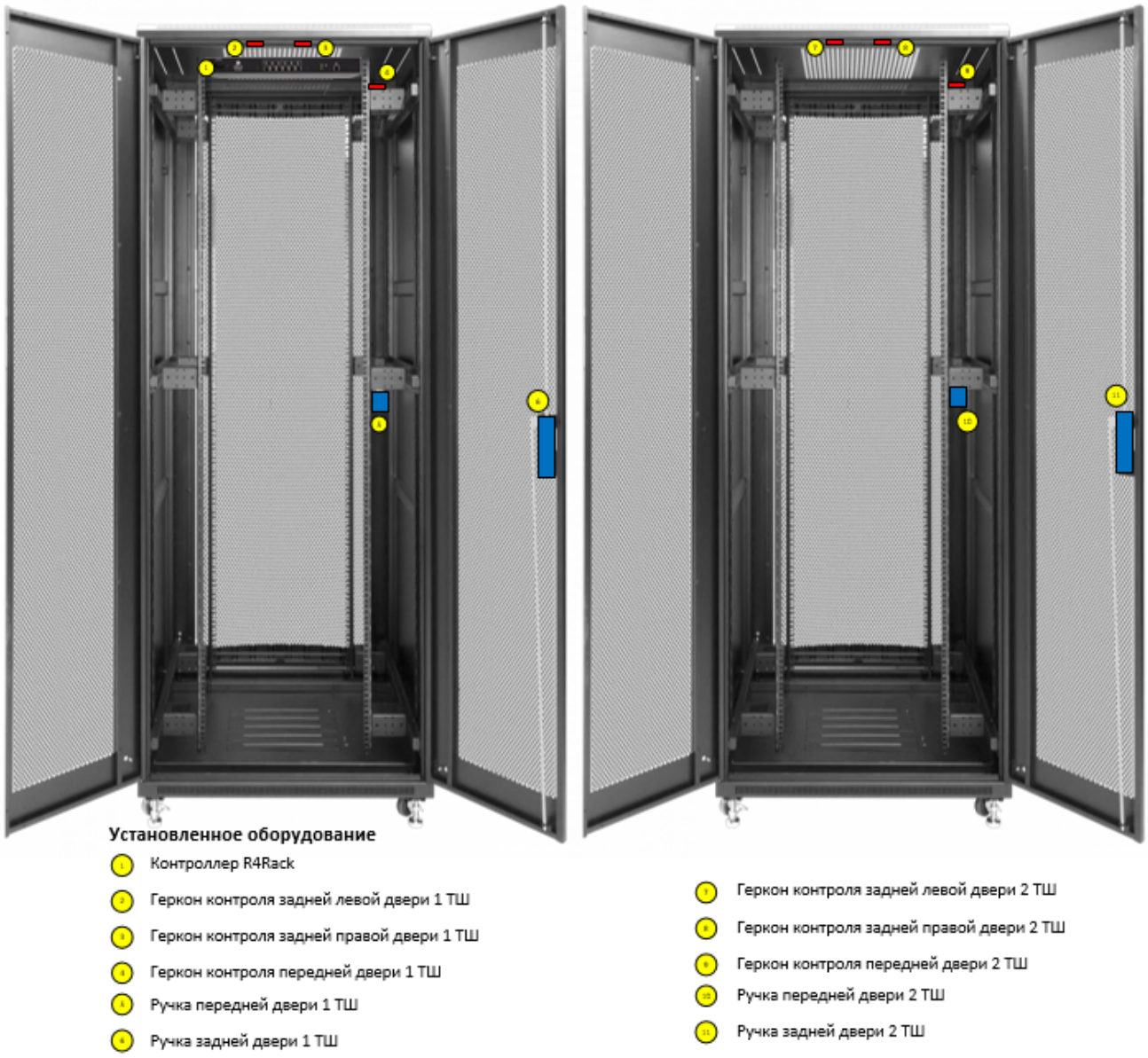
Для подключения магнитного датчика контроля двери используется кабель Castle R2R-switch-5**, поставляемый вместе с магнитным датчиком. Внешний вид кабеля изображен на рисунке.



Кабель Castle R2R-switch-5 для подключения магнитного датчика двери

** В базовой поставке удлинительные кабели Castle R2R-latch-5 и Castle R2R-switch-5 имеют длину 5 метров и не обжимаются разъемом RJ-45 (разъемы поставляется в комплекте с кабелями). Это сделано для удобства прокладки кабелей внутри конструкции шкафа, в зависимости от места и

способа установки контроллера R2Rack. Кабели после прокладки подводятся к месту установки контроллера, обрезаются до необходимой длины и оконечиваются разъемом RJ- 45 в соответсвии с рисунком распиновки соответсвенно стандатрту EIA/TIA-568B.



Типовая схема установки и подключения контроллера R4Rack (для 2-х шкафов).

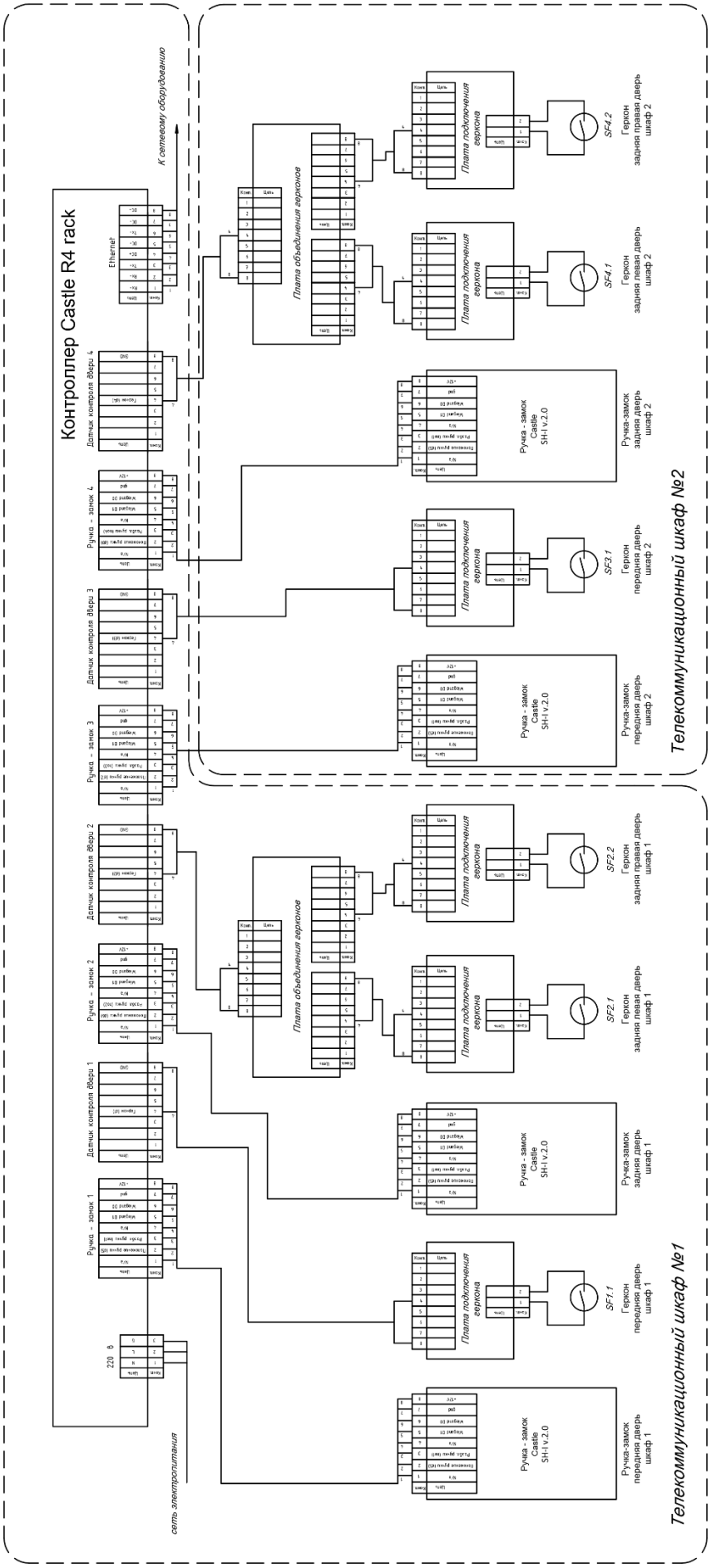


Схема подключения контроллера Castle R4Rack при использовании ручек Castle SH-I v.2.0

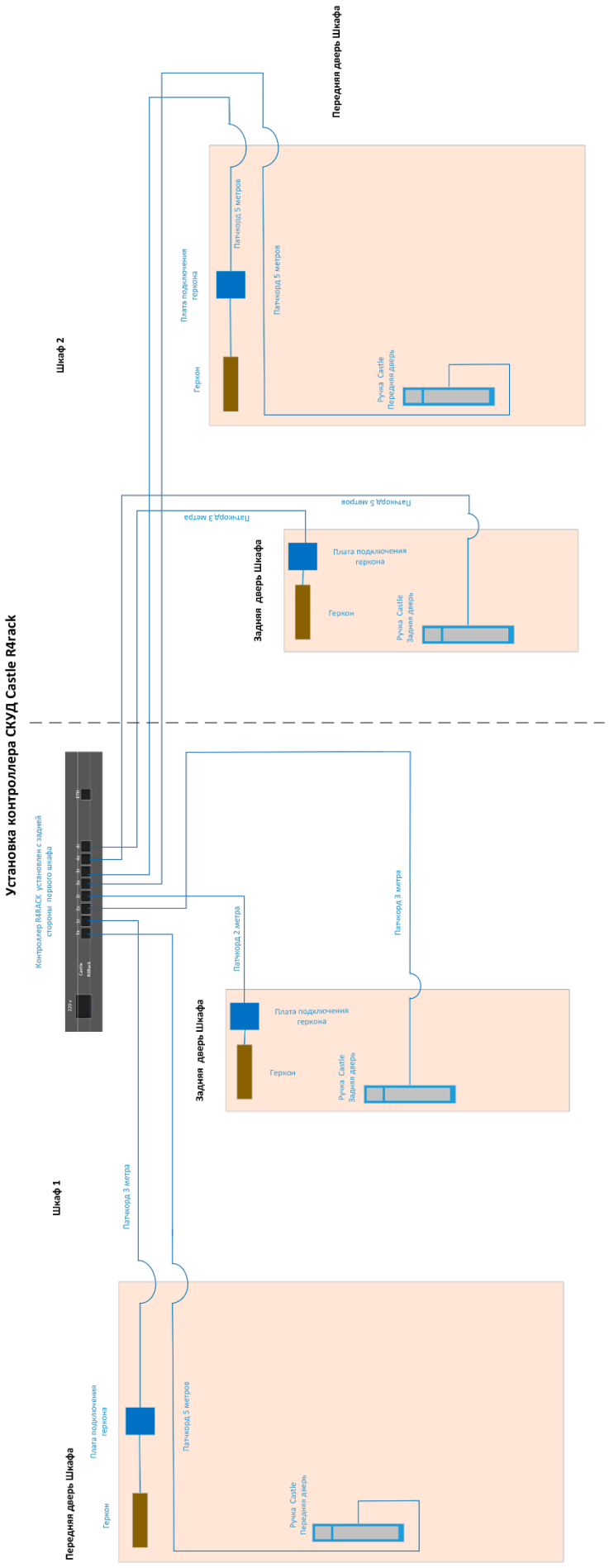


Схема коммутации контроллера Castle R4Rack в шкафу

Приложение №1

```
SIGUR DEFINITIONS ::= BEGIN

   IMPORTS
      enterprises, OBJECT-TYPE FROM SNMPv2-SMI
      TimeStamp, DateAndTime FROM SNMPv2-TC;

   sigurProduct OBJECT IDENTIFIER ::= { enterprises 56627 }

   controllers OBJECT IDENTIFIER ::= { sigurProduct 1 }

   informs OBJECT IDENTIFIER ::= { controllers 0 }

   fireAlarmInform NOTIFICATION-TYPE
      STATUS current
      DESCRIPTION
         "Fire"
      ::= { informs 1 }

   voltageWrongInform NOTIFICATION-TYPE
      STATUS current
      DESCRIPTION
         "Voltage"
      ::= { informs 2 }

   tamperStateInform NOTIFICATION-TYPE
      STATUS current
      DESCRIPTION
         "Tamper"
      ::= { informs 3 }

   batteryStateInform NOTIFICATION-TYPE
      STATUS current
      DESCRIPTION
         "Battery"
      ::= { informs 4 }

   breakInInform NOTIFICATION-TYPE
      STATUS current
      DESCRIPTION
         "Break-in"
      ::= { informs 5 }

   osdpReaderStateFail NOTIFICATION-TYPE
      STATUS current
      DESCRIPTION
         "OSDP readers"
      ::= { informs 6 }
```

```
--------------------------------------------------------------------------------------------------
   serialNumber OBJECT-TYPE
      SYNTAX OCTET STRING (SIZE (0..128))
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION
         "Serial number of controller"
      ::= { controllers 1 }
--------------------------------------------------------------------------------------------------
   osdpReaderStateTable OBJECT-TYPE
      SYNTAX SEQUENCE OF OsdpReaderStateEntry
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION
         "OSDP readers table"
      ::= { controllers 2 }

   osdpReaderStateEntry OBJECT-TYPE
      SYNTAX OsdpReaderStateEntry
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION
         "Description."
      INDEX { osdpNumber }
      ::= { osdpReaderStateTable 1 }

   OsdpReaderStateEntry ::=
      SEQUENCE {
         osdpNumber
            Counter,
         osdpState
            INTEGER,
         osdpAddress
            Counter
      }
   osdpNumber OBJECT-TYPE
      SYNTAX Counter
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION
         "Number of readers"
      ::= { osdpReaderStateEntry 1 }

   osdpState OBJECT-TYPE
      SYNTAX INTEGER {
         Unknown(0),
         NotConfigured(1),
         Misconfigured(2),
         Offline(3),
```

```
            Online(4),
            Unencrypted(5)
        }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "States"
        ::= { osdpReaderStateEntry 2 }

    osdpAddress OBJECT-TYPE
        SYNTAX Counter
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Reader address"
        ::= { osdpReaderStateEntry 3 }

-----------------------------------------------------------------------------------------------
    voltage OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Power voltage"
        ::= { controllers 3 }

    fireAlarmState OBJECT-TYPE
        SYNTAX INTEGER {
            Unknown(0),
            Fire(1),
            NotFire(2)
        }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Fire alarm"
        ::= { controllers 4 }

    localDateTime OBJECT-TYPE
        SYNTAX DateAndTime
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Local date time"
        ::= { controllers 5 }

    batteryOperation OBJECT-TYPE
        SYNTAX INTEGER {
            Unknown(0),
```

```
            Charging(1),
            EmergencyPower(2)
        }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Battery operation"
        ::= { controllers 6 }

    tamperState OBJECT-TYPE
        SYNTAX INTEGER {
            Unknown(0),
            Normal(1),
            BreakIn(2)
        }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Tamper state"
        ::= { controllers 7 }

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

    ioPortStateTable OBJECT-TYPE
        SYNTAX SEQUENCE OF IoPortStateEntry
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "IO Ports"
        ::= { controllers 8 }

    ioPortStateEntry OBJECT-TYPE
        SYNTAX IoPortStateEntry
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Description."
        INDEX { accessPoint, function }
        ::= { ioPortStateTable 1 }

    IoPortStateEntry ::=
        SEQUENCE {
            accessPoint
                INTEGER,
            function
                INTEGER,
            physicalPin
                INTEGER,
            portState
```

```
            INTEGER,
        direction
            INTEGER
    }

accessPoint OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Access point"
    ::= { ioPortStateEntry 1 }

function OBJECT-TYPE
    SYNTAX INTEGER {
        Unknown(0),
        liTurnstilePanelA(1),
        liTurnstilePanelB(2),
        liTurnstilePanelL(3),
        loTurnstileIndA(4),
        loTurnstileIndB(5),
        loTurnstileIndL(6),
        liTurnstileDetA(7),
        liTurnstileDetB(8),
        loTurnstileCntlA(9),
        loTurnstileCntlB(10),
        loTurnstileCntlL(11),
        liDoorDet(12),
        liDoorRteA(13),
        liDoorRteB(14),
        liDoorRteX(15),
        liDoorLock(16),
        loDoorLock(17),
        loDoorUnlock(18),
        liGateDetA(19),
        liGateDetB(20),
        liGateDetC(21),
        liGatePanelM(22),
        liGatePanelS(23),
        loOprAllowed(24),
        loOprDeny(25),
        liFire(26),
        liOpd(27),
        loBreakAlarm(28),
        liRegDetA(29),
        liRegDetB(30),
        liTurnstileDetX(31),
        loImpAllowA(32),
        loImpAllowB(33),
```

```
        loImpDenyA(34),
        loImpDenyB(35),
        liReqmngstateNormal(36),
        liReqmngstateLock(37),
        liReqmngstateUnlock(38),
        loAlmNormal(39),
        loAlmAlarm(40),
        loDoorHoldAlarm(41),
        liDcin(42),
        loMngstateLock(43),
        loAcceptA(44),
        loAcceptB(45),
        loRejectA(46),
        loRejectB(47),
        loMngstateUnlock(48),
        loPowerMain(49),
        loPowerStandby(50),
        loTraflightA(51),
        loTraflightB(52),
        loCardinpocket(53),
        loLedc(54),
        loWaitingAlkoA(55),
        loWaitingAlkoB(56),
        liSurpressalko(57),
        liHallsensor(58),
        loWaitingEscortA(59),
        loWaitingEscortB(60),
        loGateOpen(61),
        loGateClose(62),
        loGateStop(63),
        loGateOpen2(64),
        loGateClose2(65),
        liGateDd(66),
        liGateDu(67),
        liResetPeopleCnt(68),

        liReaderOut(-1),
        liReaderIn(-2),
        liReaderUn(-3)
    }
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Function"
    ::= { ioPortStateEntry 2 }

physicalPin OBJECT-TYPE
    SYNTAX INTEGER{
        portDet1Pass1(0),
```

```
        portDet2Pass2(1),
        portDet3Rte1(2),
        portDet4Rte2(3),
        portDet5StopPass3(4),
        portDet6Pass4(5),
        portDet7Rte3(6),
        portDet8Rte4(7),
        portDet9Auxin1(8),
        portDet10Auxin2(9),
        portOpd(10),
        portFire(11),
        portDcd(12),
        portK1(13),
        portK2(14),
        portK3(15),
        portK4(16),
        portOut1Auxout1(17),
        portOut2Auxout2(18),
        portOut3(19),
        portOut4(20),
        portOut5(21),
        portL1ALed1(22),
        portL1B(23),
        portL2ALed2(24),
        portL2B(25),
        portL3A(26),
        portL3B(27),
        portL4A(28),
        portL4B(29),
        portCpi1(30),
        portCpi2(31),
        portCpi3(32),
        portLedrx(33),
        portLedtx(34),
        portLedpwr(35),
        portSnd(36),
        portRst(37),
        portBat(38),

        port1(-1),
        port2(-2),
        port3(-3),
        port4(-4)
    }
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "physicalPin"
    ::= { ioPortStateEntry 3 }
```

```
    portState OBJECT-TYPE
        SYNTAX INTEGER {
            Unknown(0),
            Inactive(1),
            Active(2)
        }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "States"
        ::= { ioPortStateEntry 4 }

    direction OBJECT-TYPE
        SYNTAX INTEGER {
            Unknown(0),
            Input(1),
            Output(2)
        }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Pin direction"
        ::= { ioPortStateEntry 5 }

    activeState OBJECT-TYPE
        SYNTAX INTEGER {
            Normal(0),
            Inverted(1)
        }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Active state"
        ::= { ioPortStateEntry 6 }
END
```