Fix Me! *Ведется работа над переводом*

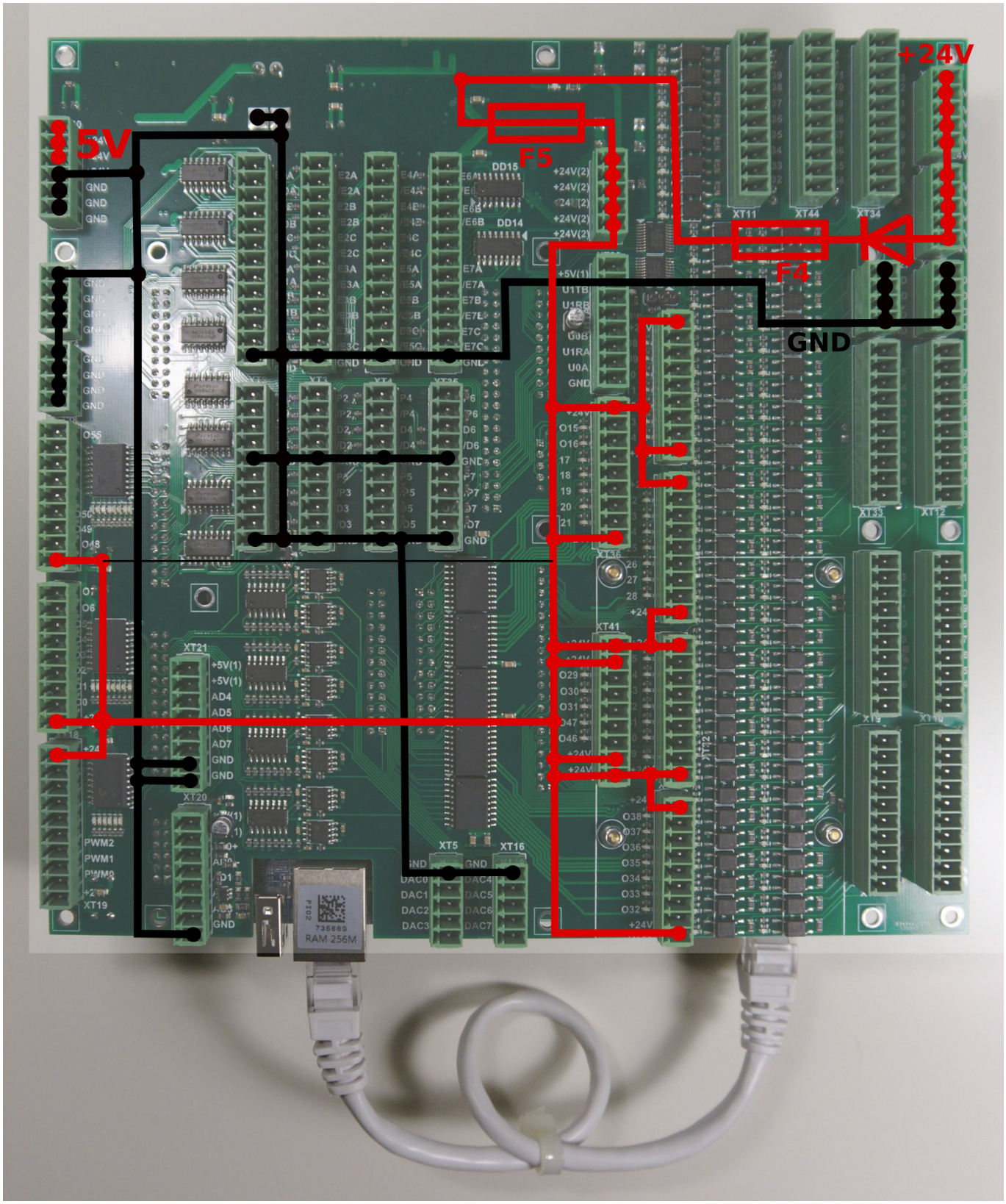
Контроллер myCNC-ET15

(Превью)

Подключение питания

Плата управления myCNC-ET15 использует 24В постоянного тока. Плата содержит 4 контакта для подключения + 24В (соединены внутри) и несколько контактов GND для удобного подключения внешних устройств. Контакты питания 24 В постоянного тока, а также контакты + 24В и GND показаны на рисунке ниже.

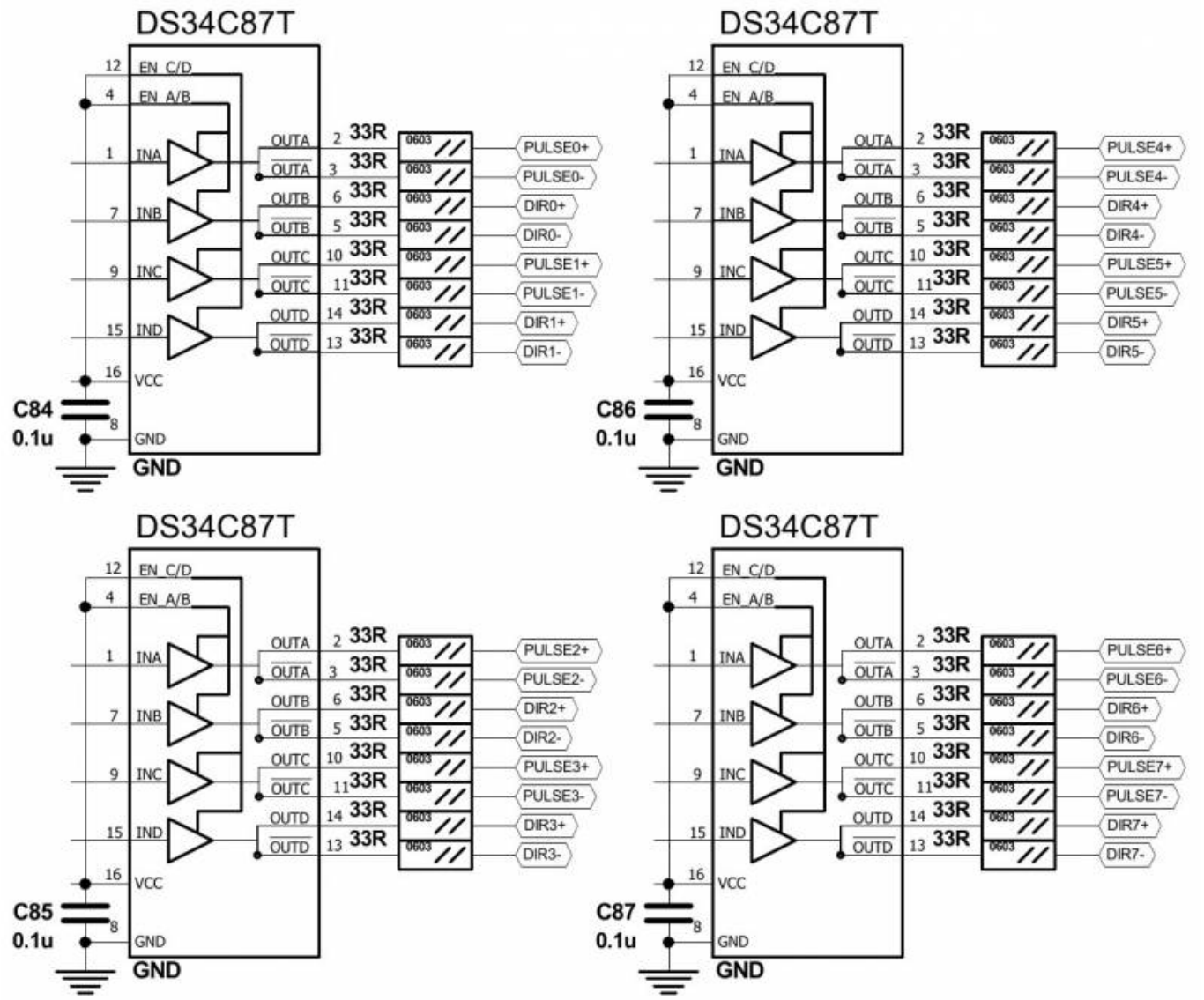
ПРИМЕЧАНИЕ: На плате есть набор неправильно маркированных выходов 5В (помеченных на плате как 24В). Ниже приведена правильная распиновка блока питания:



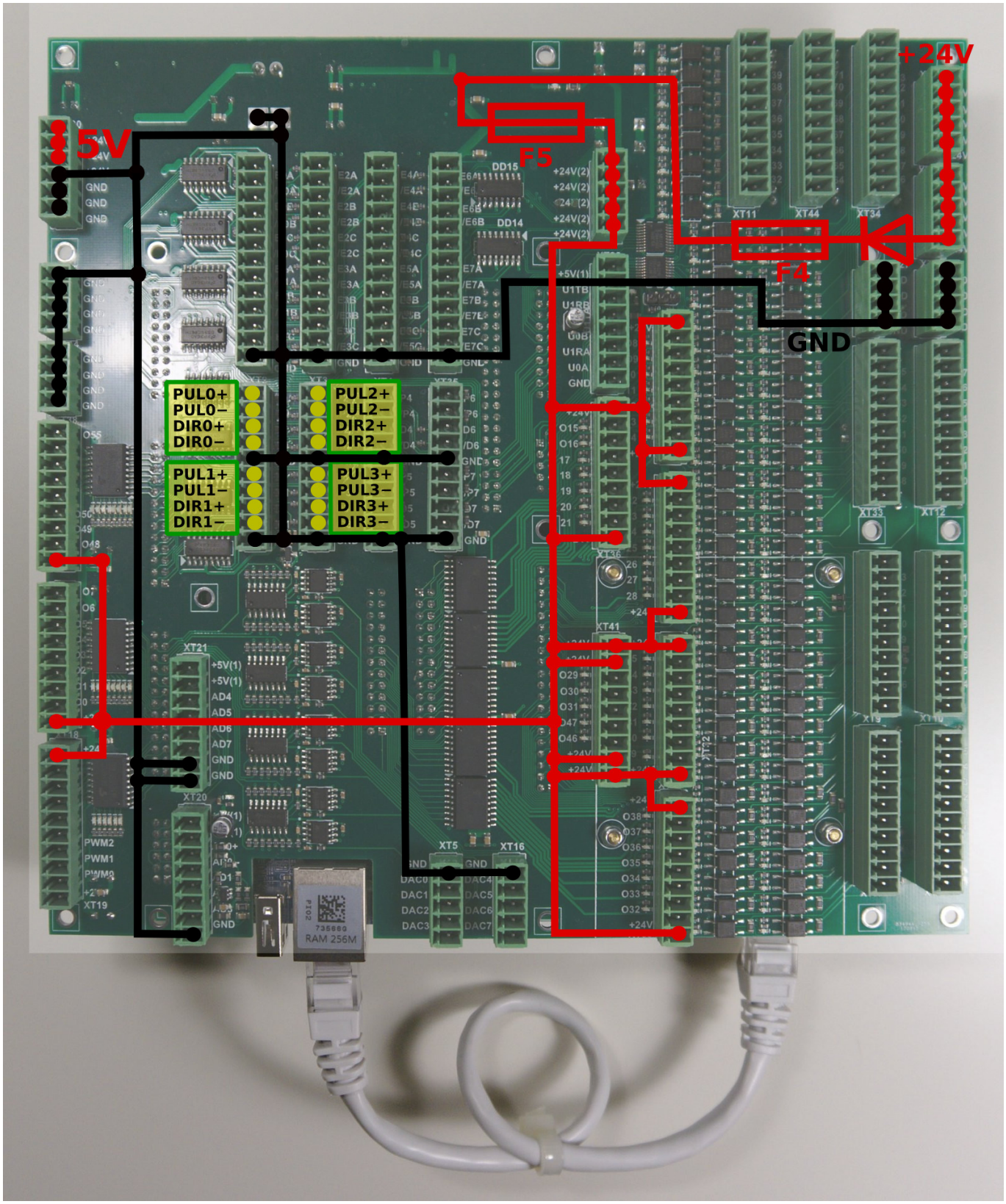
Выходы Pulse-Dir

ET15 имеет 8 выходов pulse/dir, максимальная частота импульсов 3 МГц.

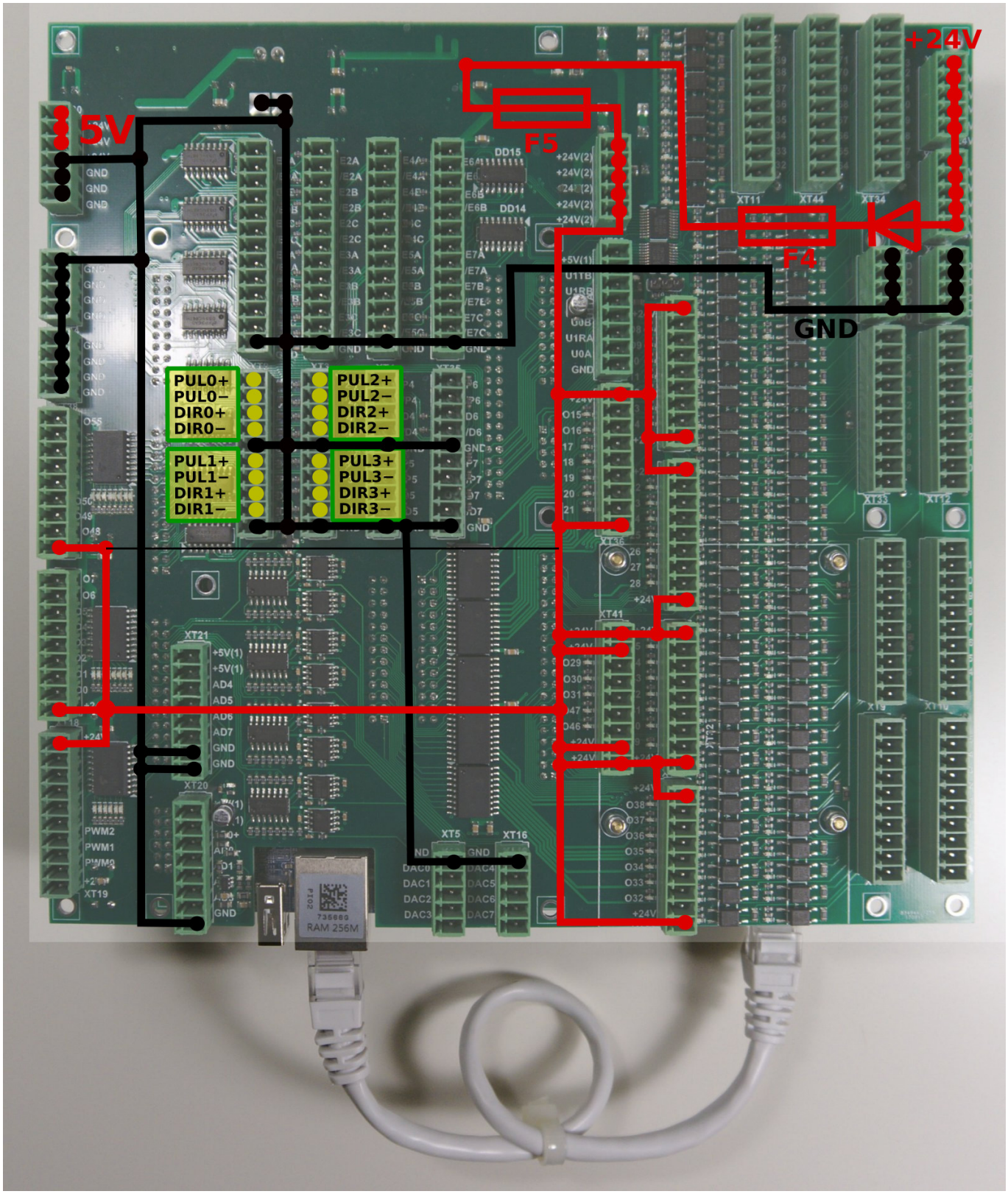
Импульсные выходы ET15 соответствуют стандарту RS422 и совместимы с большинством серво и шаговых драйверов (линейный драйвер с парафазными сигналами положительной и отрицательной полярности). Внутренняя схема для pulse-dir показана на рисунке ниже.



Каналы PULSE-DIR 0,1,2,3:



Каналы PULSE-DIR 4,5,6,7:



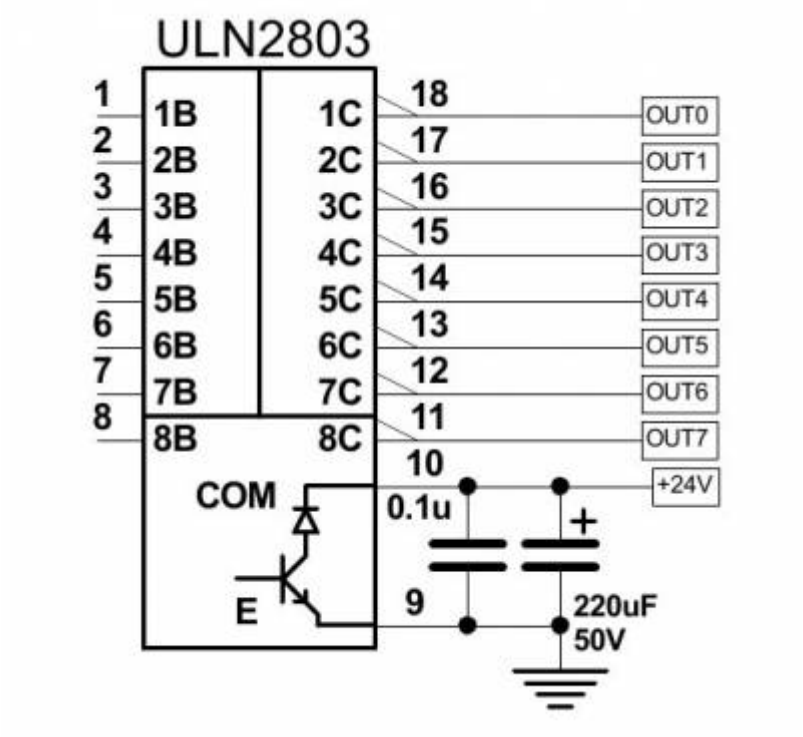
ET15 - Выходы

Плата ET15 содержит 64 выхода

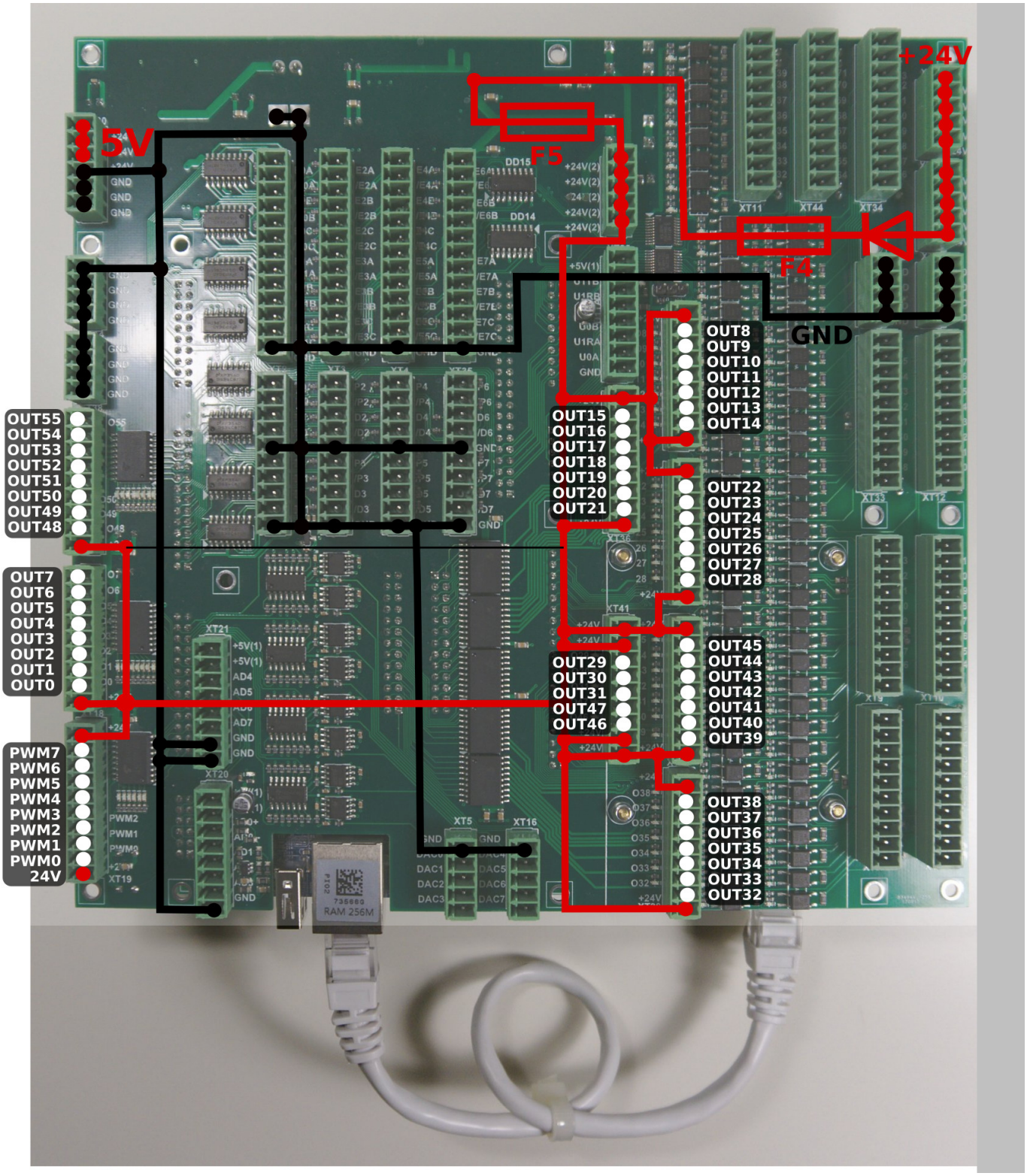
- 56 выходов с открытым коллектором (OUT # 0-OUT # 55)
- 8 выходов ШИМ (ШИМ №0 - ШИМ №7)

Внутренняя схема показана на рисунке ниже. Чип матрицы транзисторов Дарлингтона

ULN2803 используется для буферизации двоичных выходов в ET15. Каждый чип содержит 8 транзисторов и обрабатывает 8 двоичных выходов.

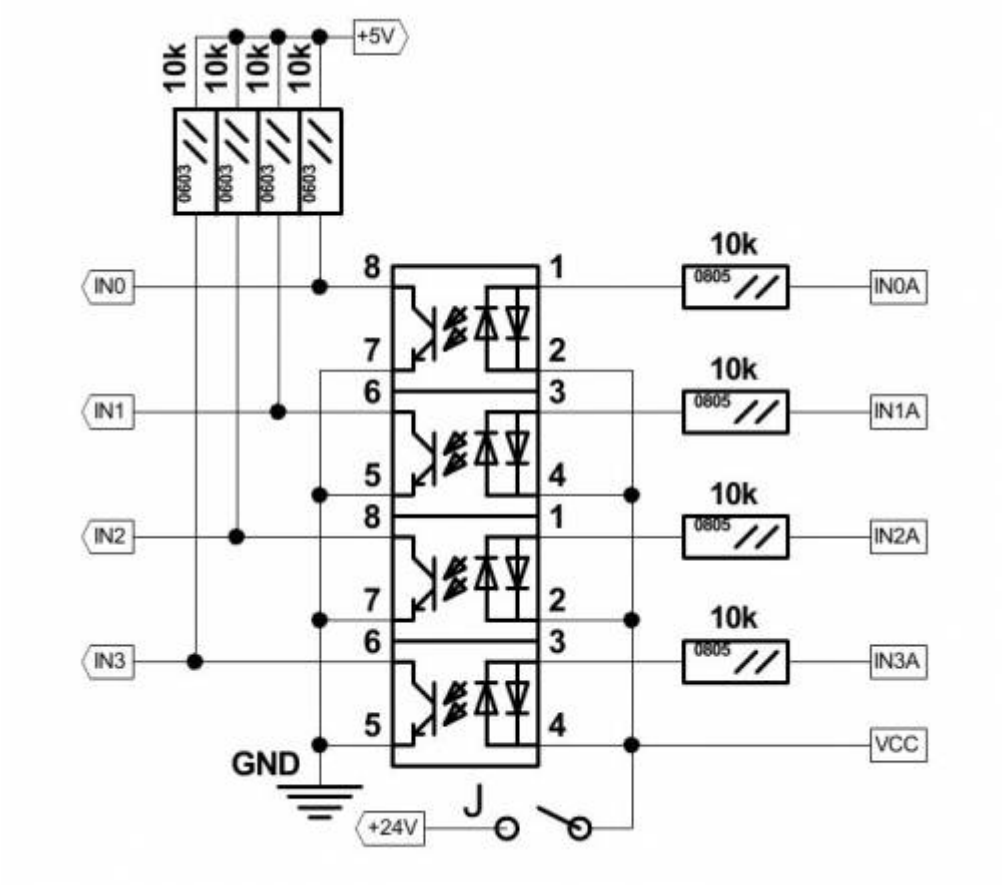


Выходы на плате ET15:



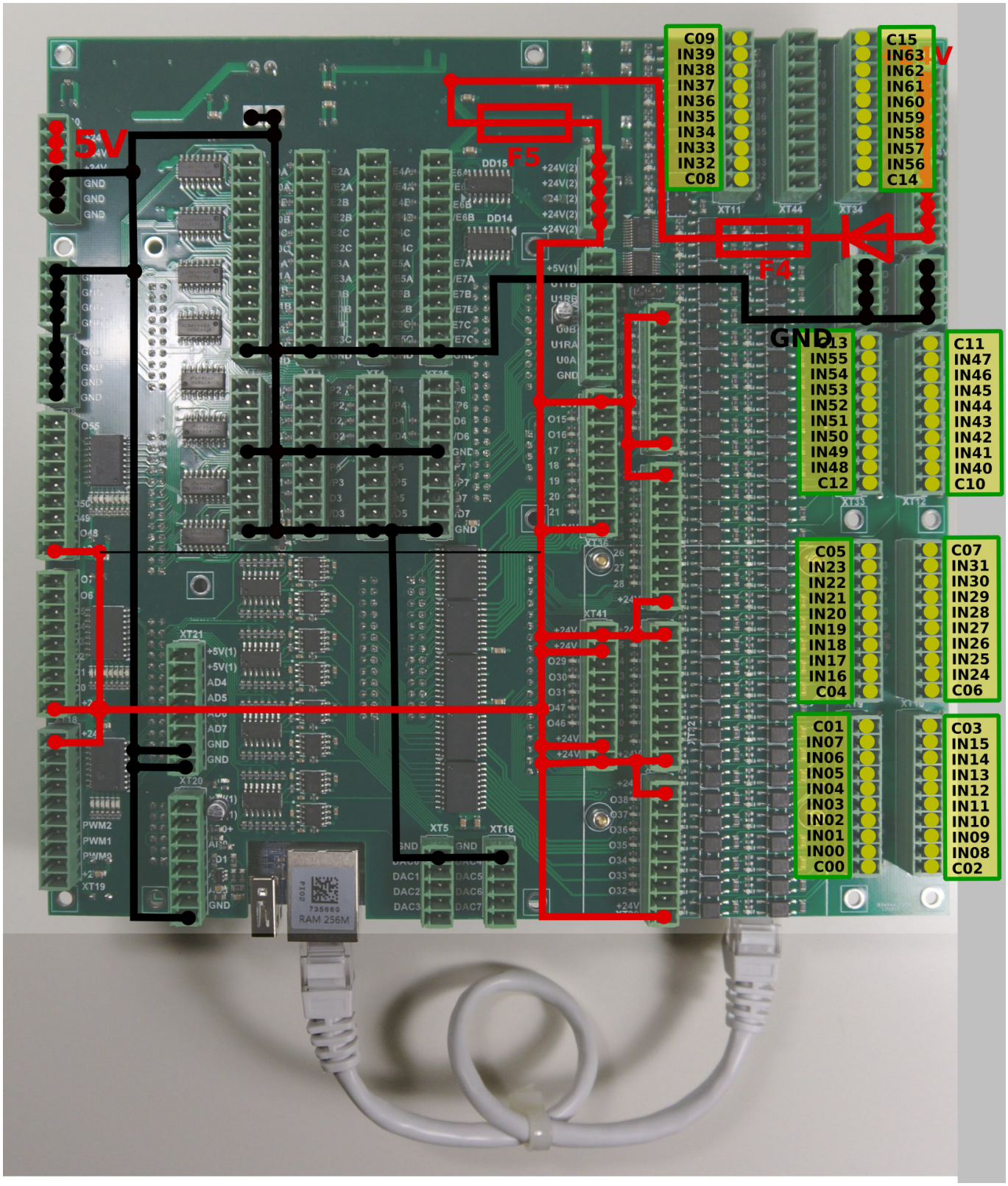
Гальванически развязанные входы

Плата управления ET15 имеет 64 гальванически развязанных двоичных входов, 16 + 2 группы по 4 входа в каждой. Каждая группа имеет отдельные контакты питания, поэтому входы могут получать питание от разных источников. Возможно одновременное использование датчиков PNP и NPN. Схема группы из четырех входов показана на рисунке ниже.



16 групп (64 контакта контакта в целом) являются независимыми входами.

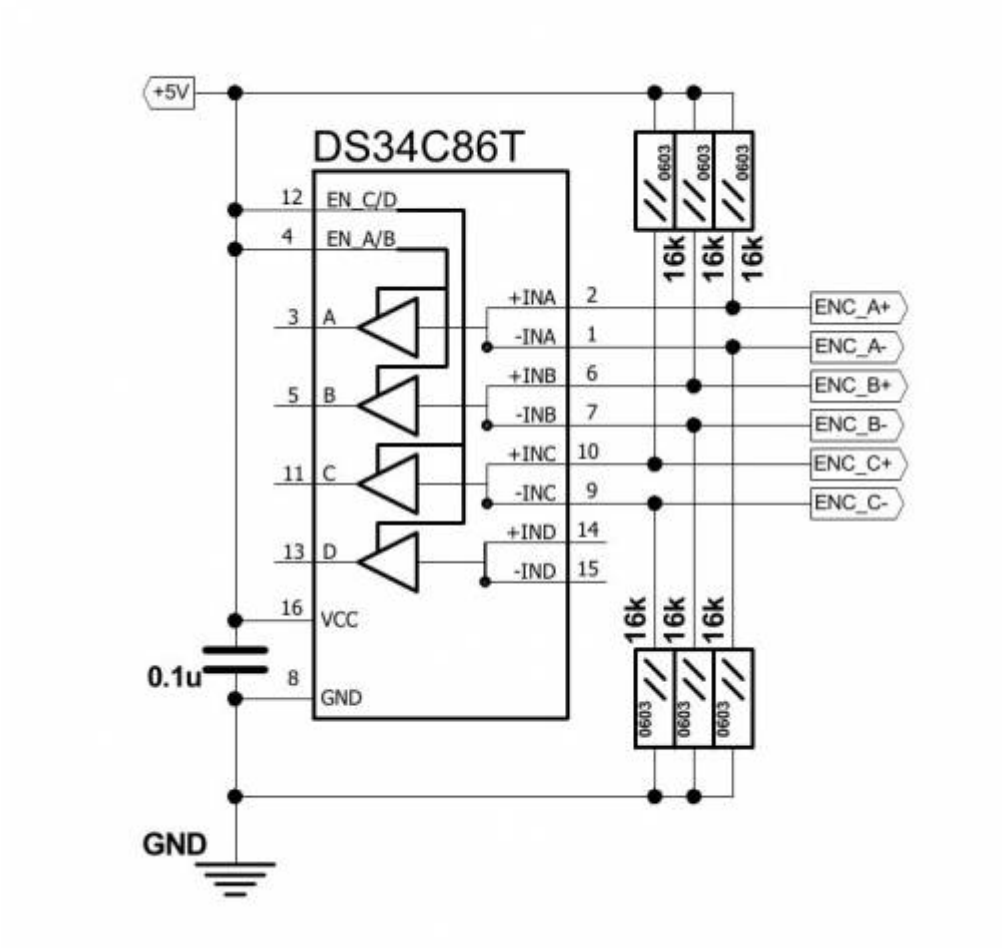
Дополнительные 2 группы (8 входов оптопары) подключены к 8и контактам входов энкодера (контакты ENCODER 5B, 5C, 6A, 6B, 6C, 7A, 7B, 7C). Измените положение SW1, чтобы выбрать либо гальванически развязанные входы, либо входы линейного драйвера для ВХОДОВ # 64 ... # 71.



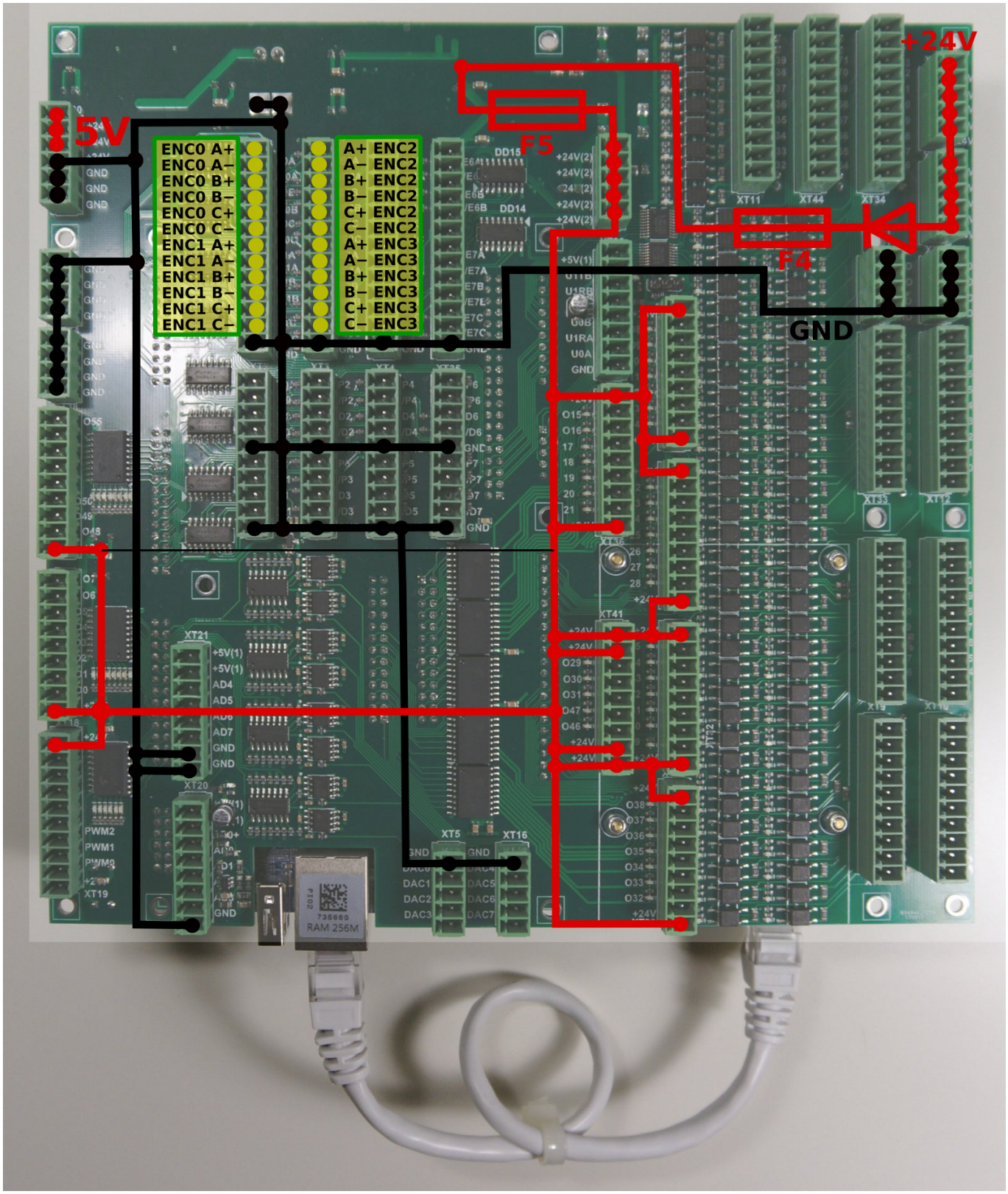
ET15 Входы энкодера

Плата ET15 имеет 8 входов инкрементного энкодера. Входы энкодера на плате ET15 соответствуют стандарту RS422 и совместимы с большинством энкодеров сервоприводов и линейных драверов. Микросхема 34C86 используется в ET15 в качестве приемника сигналов энкодера. Внутренняя схема входов энкодера показана на рисунке ниже.

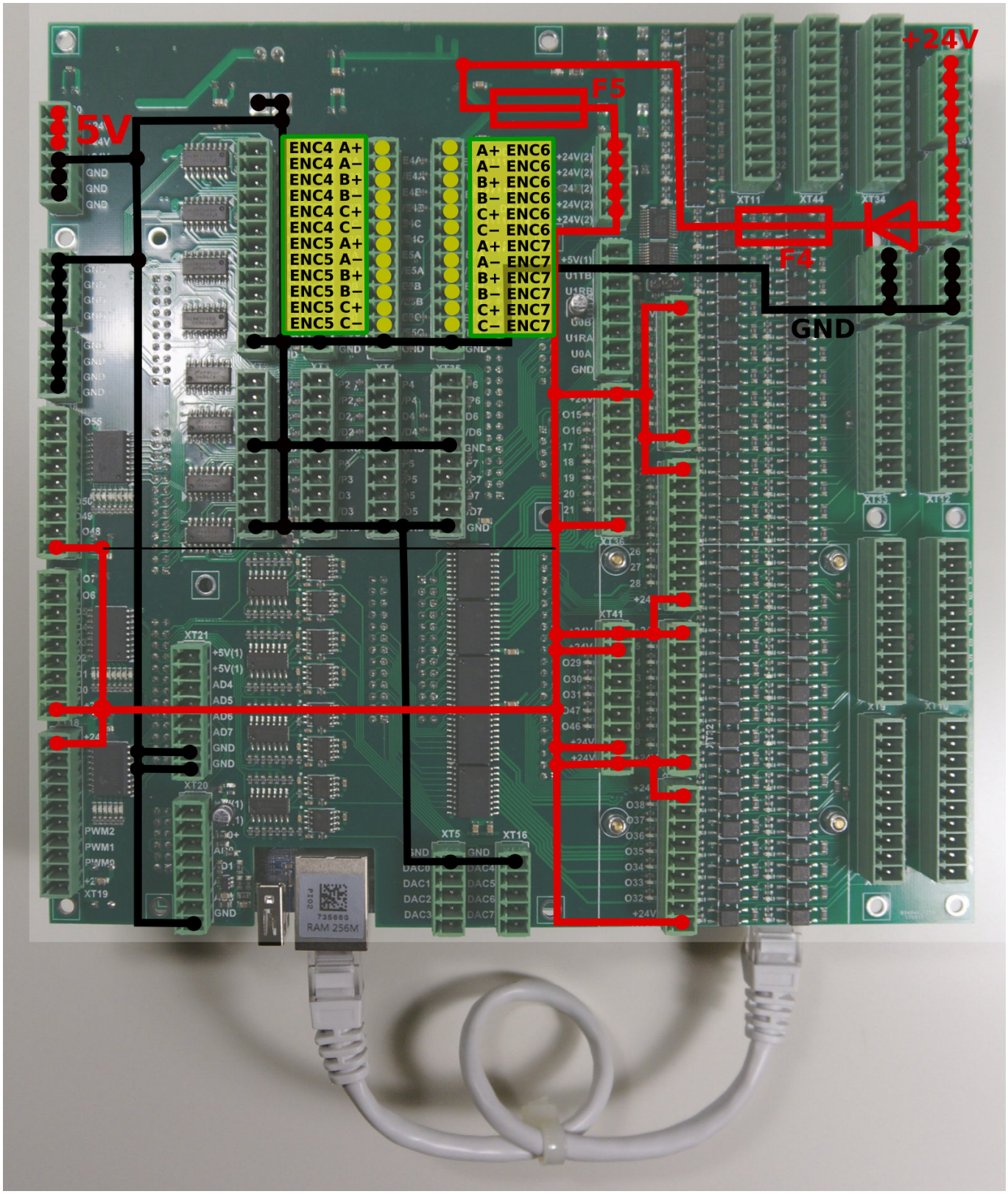
INCREMENTAL ENCODER входная схема (1 энкодер, показаны сигналы ABC)



ЭНКОДЕРЫ каналы 0,1,2,3:



ЭНКОДЕРЫ каналы 4,5,6,7:



Входы энкодера отображаются в списке входов общего назначения, зарезервированы адреса от 64 до 87. Плата ET15 может использовать до 88 двоичных входов.

Входы АЦП

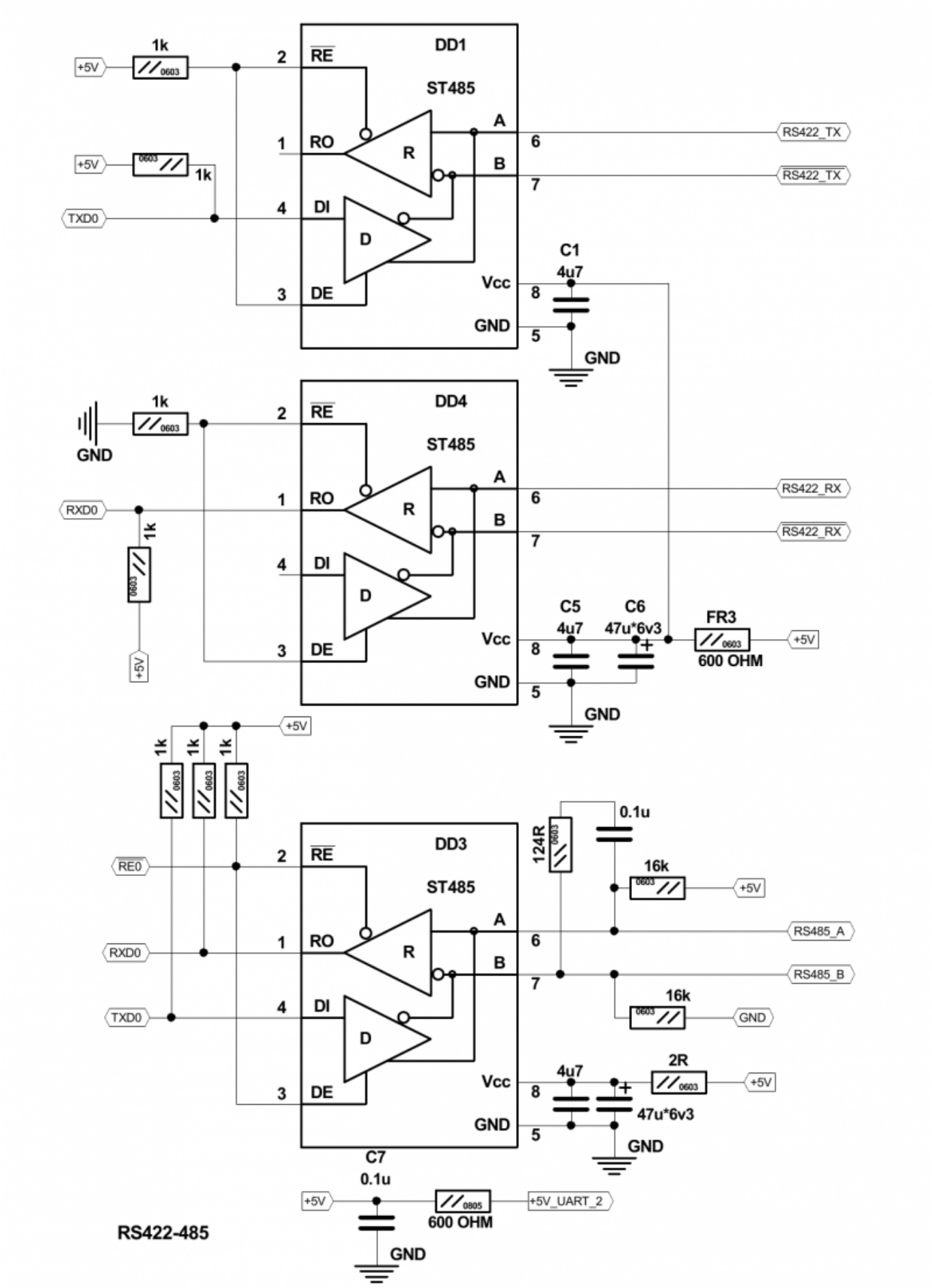
Плата управления myCNC-ET15 имеет 8 входов АЦП в диапазоне 0 ... 5 В. Входные разъемы АЦП также имеют разъемы GND и + 5В постоянного тока для удобного подключения

потенциометра. На рисунке ниже показан пример подключения потенциометра ко входу ADC2.

Шина RS422 / RS485

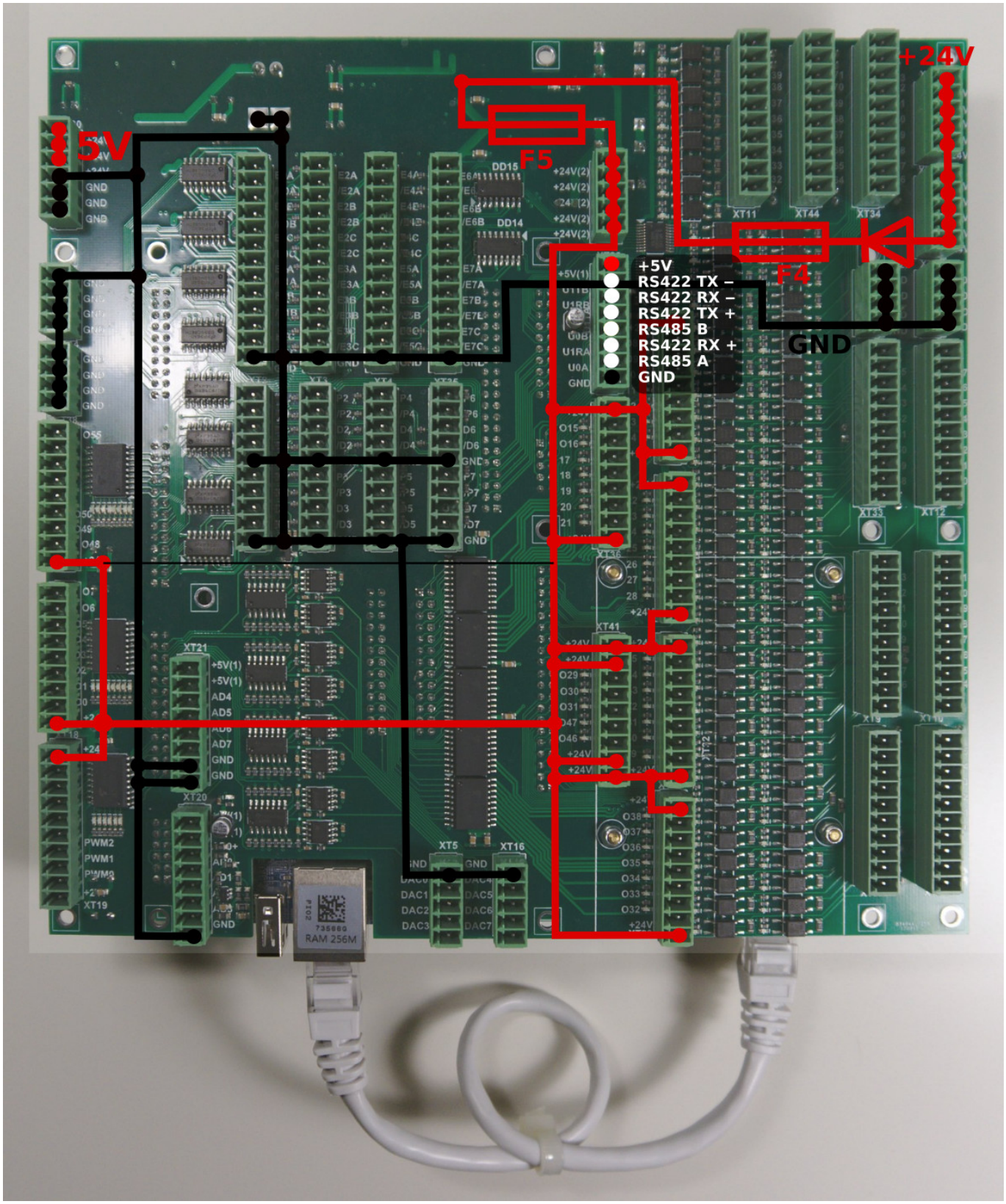
Интерфейсы шины RS422 и RS485 реализованы в аппаратном обеспечении платы управления myCNC-ET15. Включены интерфейсы Modbus ASCII/RTU и Hypertherm Serial для RS485 и RS422.

Схемы выходов для интерфейсов RS422/RS485 приведены ниже.



Плата управления myCNC-ET15 имеет разъем для шины RS422/RS485. Схема контактов разъема

для RS422/RS485 показана ниже



Пример настройки шпинделя соединенного по Modbus RS485

Настройка шпинделя на Modbus RS485

Примеры подключения

Пример подключения трехпроводного датчика NPN

```
Перемычки J1, J2, J3, J4 открыты.
```

Пример подключения трехпроводного датчика PNP

```
Перемычки J1, J2, J3, J4 открыты.
```

Пример подключения переключателя

```
Перемычка для выбранной группы (J1, J2, J3, J4) закрыта.
```

Общий провод для 4-х оптопар подключается к внутреннему +24 В, если перемычка замкнута. Переключатель должен замкнуть другой вход оптопары на GND (0 В), чтобы активировать входной контакт.

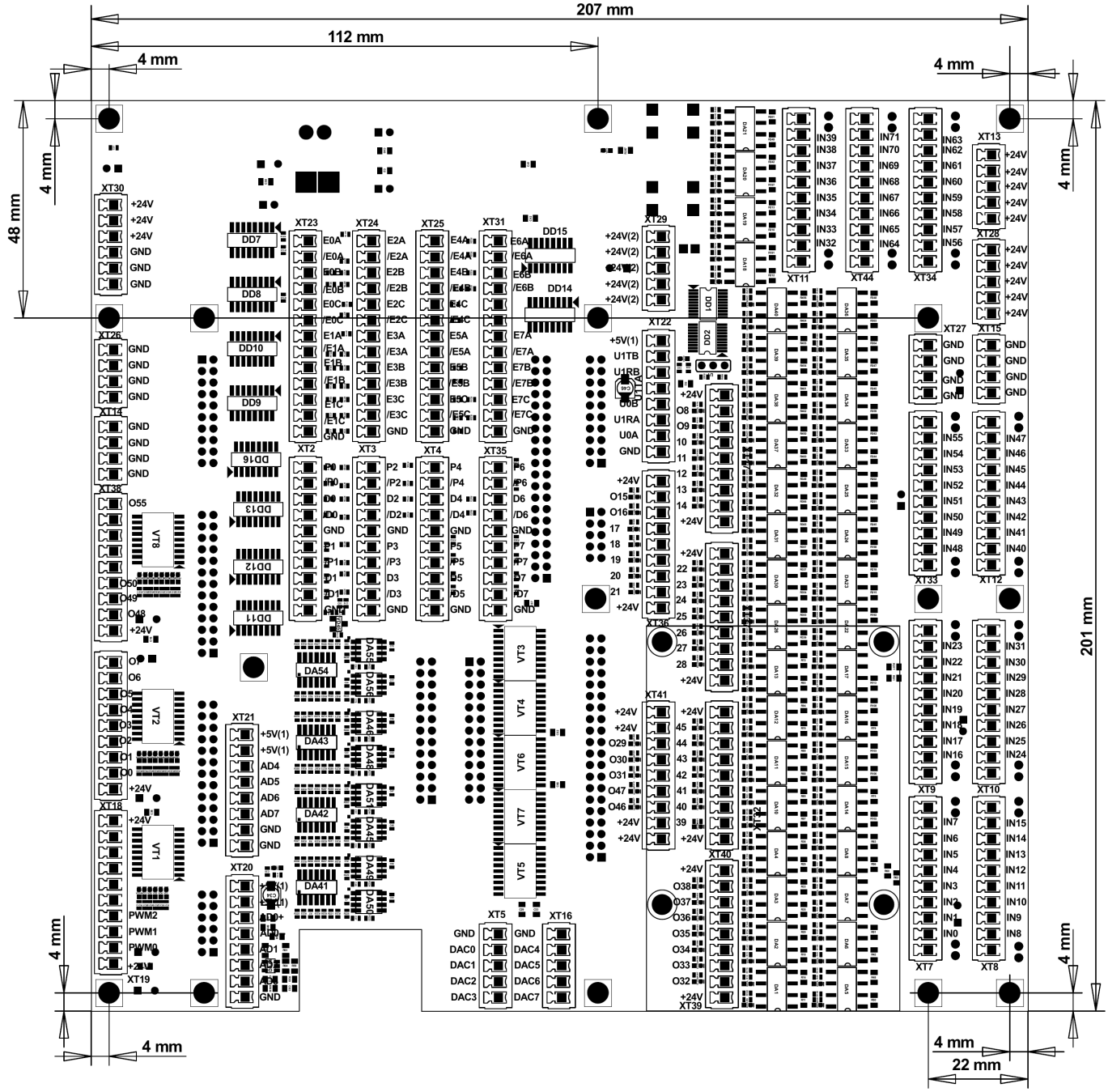
J4 должен быть закрыт, чтобы подключить контакт оптопары к +24В. Переключатель должен замыкать провод на GND (0 В).

Управление скоростью шпинделя через выход ЦАП (0-10 В)

Размеры платы

PDF: http://cnc42.com/downloads/et15bb-r09.pdf

DXF: http://cnc42.com/downloads/et15bb-r09.dxf



SSH доступ к ET15

Встроенное программное обеспечение платы ET15 сделано на основе RT-Linux и SSH-сервер и может быть настроено для получения доступа к плате, настройкам, и возможностям изменения и обновления встроенного программного обеспечения.

Данные для доступа:

порт SSH	22
логин	mycnc
пароль	operator
команда для доступа	ssh mycnc@192.168.0.69
команда для доступа	ssh mycnc@192.168.1.69

команда для доступа	ssh mycnc@192.168.4.69

Кинематика осей для ET15

Встроенное ПО платы ET15 поддерживает преобразование кинематики, определяемое пользователем.

Прошивка ET15 запускает процедуру **MKinematics** (запускатеся каждый серво цикл).

Код процедуры:

```
void MKinematics(int64_t * input, int64_t * output, uint32_t naxes);
```

Предполагается, что процедура заполняет массив выходных координат в соответствии с кинематикой машины. Процедура MKinematics может содержать простое присвоение выходных координат входным значениям, если нет необходимости в кинематических преобразованиях.

```
void KinematicsPlugin::MKinematics(int64_t *input, int64_t *output, uint32_t
naxes)
{
    for (uint32_t i=0;i<naxes;i++) output[i]=input[i];
    return;
}
```

Альтернативный способ полностью отключить плагин кинематики - это удалить файл библиотеки плагинов **libkinematicsplugin.so** из папки плагинов.

Пример процедуры MKinematics для робототехники показан ниже.

```
void KinematicsPlugin::MKinematics(int64_t *input, int64_t *output, uint32_t
naxes)
{

    double x=input[0]*InputRatio[0];
    double y=input[1]*InputRatio[1];
    double z=input[2]*InputRatio[2];

    double M2=x*x+y*y;
    double M=sqrt(M2);
    double L2=M2+z*z;
    double L=sqrt(L2);

    double a=acos(x/M);
    double b=asin(z/L);
    double d=acos((R1*R1+R2*R2-L2)/(2*R1*R2));
    double f=asin(sin(d)*R2/L);
    double c=b+f;

    output[0]=a*OutputRatio[0];
    output[1]=c*OutputRatio[1];
```

```
    output[2]=d*OutputRatio[2];

}
```

Процедура MKinematics является частью полного класса C ++ **MKinematicsPlugin**, который может иметь другие переменные и функции помимо основного **MKinematics**.

Например, MKinematics использует переменные R1, R2, которые определяют длину соединений. Значения переменных могут быть определены статически в конструкторе класса

```
KinematicsPlugin::KinematicsPlugin(QObject *parent) :
    QObject(parent)
{
    for (int i=0;i<32;i++)
    {
        InputRatio[i]=1;
        OutputRatio[i]=1;
    }
    R1=100;
    R2=50;
}
```

или могут назначаться из кода прошивки путем запуска процедуры **setParameters**, которая является частью интерфейса плагина

```
void KinematicsPlugin::setParameters(uint32_t addr, double param)
{
    switch (addr)
    {
    case 0: R1=param;break;
    case 1: R2=param;break;
    }
}
```

Программное обеспечение контроллера myCNC использует 64-битные значения с фиксированной точкой и управляет «единицами», которые равны «импульсам» (для драйверов двигателя с импульсным приводом) или «единице энкодера» для аналогового сервоуправления.

Для расчета кинематических формул, значения должны быть в реальных единицах, таких как миллиметр, дюйм, градус или радиан.

Соотношения для перевода «импульсов» в «миллиметры» или «радианы» до перевода кинематики можно назначать статически или получать из основного программного обеспечения через процедуру **setInputRatios**

```
void KinematicsPlugin::setInputRatios(double * ratio, uint32_t naxes)
{
    for (uint32_t i=0;i<naxes;i++) InputRatio[i]=ratio[i];
}
```

После перевода кинематики значения координат следует переводить обратно в «импульсные» единицы. Это можно сделать с помощью процедуры **setOutputRatios** или назначить статически в коде плагина.

```
void KinematicsPlugin::setOutputRatios(double * ratio, uint32_t naxes)
{
    for (uint32_t i=0;i<naxes;i++) OutputRatio[i]=ratio[i];
}
```

Плагин для кинематики находится на стадии разработки. В будущем возможны изменения.

Полный пример плагина кинематики:

Kinematics plugin source file

kinematicsplugin.cpp

```
#include "kinematicsplugin.h"
#include <math.h>
KinematicsPlugin::KinematicsPlugin(QObject *parent) : QObject(parent)
{
    for (int i=0;i<32;i++)
    {
        InputRatio[i]=1;
        OutputRatio[i]=1;
    }
    R1=100;
    R2=50;
}

void KinematicsPlugin::setParameters(uint32_t addr, double param)
{
    switch (addr)
    {
    case 0: R1=param;break;
    case 1: R2=param;break;
    }
}

void KinematicsPlugin::MKinematics(int64_t *input, int64_t *output,
uint32_t naxes)
{
    for (uint32_t i=0;i<naxes;i++) output[i]=input[i];

//    for (uint32_t i=0;i<naxes;i++) output[i]=input[0]/(i+1);
    return;

    for (uint32_t i=0;i<naxes;i++)
output[i]=input[i]*InputRatio[i]*OutputRatio[i];

    double x=input[0]*InputRatio[0];
```

```
    double y=input[1]*InputRatio[1];
    double z=input[2]*InputRatio[2];

    double M2=x*x+y*y;
    double M=sqrt(M2);
    double L2=M2+z*z;
    double L=sqrt(L2);

    double a=acos(x/M);
    double b=asin(z/L);
    double d=acos((R1*R1+R2*R2-L2)/(2*R1*R2));
    double f=asin(sin(d)*R2/L);
    double c=b+f;

    output[0]=a*OutputRatio[0];
    output[1]=c*OutputRatio[1];
    output[2]=d*OutputRatio[2];

}


void KinematicsPlugin::setInputRatios(double * ratio, uint32_t naxes)
{
    for (uint32_t i=0;i<naxes;i++) InputRatio[i]=ratio[i];
}

void KinematicsPlugin::setOutputRatios(double * ratio, uint32_t naxes)
{
    for (uint32_t i=0;i<naxes;i++) OutputRatio[i]=ratio[i];
}
```

Дополнительно необходим следующий файл:

kinematicsplugin.h

```
#ifndef KINEMATICSPLUGIN_H
#define KINEMATICSPLUGIN_H
#include "kinematicsinterface.h"
#include <QObject>
#include <QtPlugin>
class KinematicsPlugin : public QObject, KinematicsInterface
{
    Q_OBJECT
    Q_PLUGIN_METADATA(IID Kinematics_Interface_iid FILE
"kinematicsplugin.json")
    Q_INTERFACES(KinematicsInterface)
public:
    KinematicsPlugin(QObject *parent = 0);
    void MKinematics(int64_t * input, int64_t * output, uint32_t
naxes);
```

```
    void setInputRatios(double * ratio, uint32_t naxes);
    void setOutputRatios(double * ratio, uint32_t naxes);
    void setParameters(uint32_t addr, double param);
protected:
    double InputRatio[32];
    double OutputRatio[32];
    double R1,R2;
};
#endif // KINEMATICSPLUGIN_H
```

Также в дополнение:

kinematicspluginterface.h

```
ifndef KINEMATICSINTERFACE_H
#define KINEMATICSINTERFACE_H
#include <stdint.h>
#include <QtPlugin>
class KinematicsInterface
{
public:
    virtual ~KinematicsInterface() {}
    virtual void MKinematics(int64_t * input, int64_t * output,
uint32_t naxes) = 0;
    virtual void setInputRatios(double * ratio, uint32_t naxes) = 0;
    virtual void setOutputRatios(double * ratio, uint32_t naxes) = 0;
    virtual void setParameters(uint32_t addr, double param) = 0;
};

#define Kinematics_Interface_iid "pv-
automation.myCNC.ET15.R1.KinematicsInterface"
Q_DECLARE_INTERFACE(KinematicsInterface, Kinematics_Interface_iid)
#endif // KINEMATICSINTERFACE_H
```

Полный архив исходников для этого примера можно скачать здесь:

http://pv-automation.com/downloads/kinematics_2018-0205_0000.tar.bz2
http://cnc42.com/downloads/1366.7z