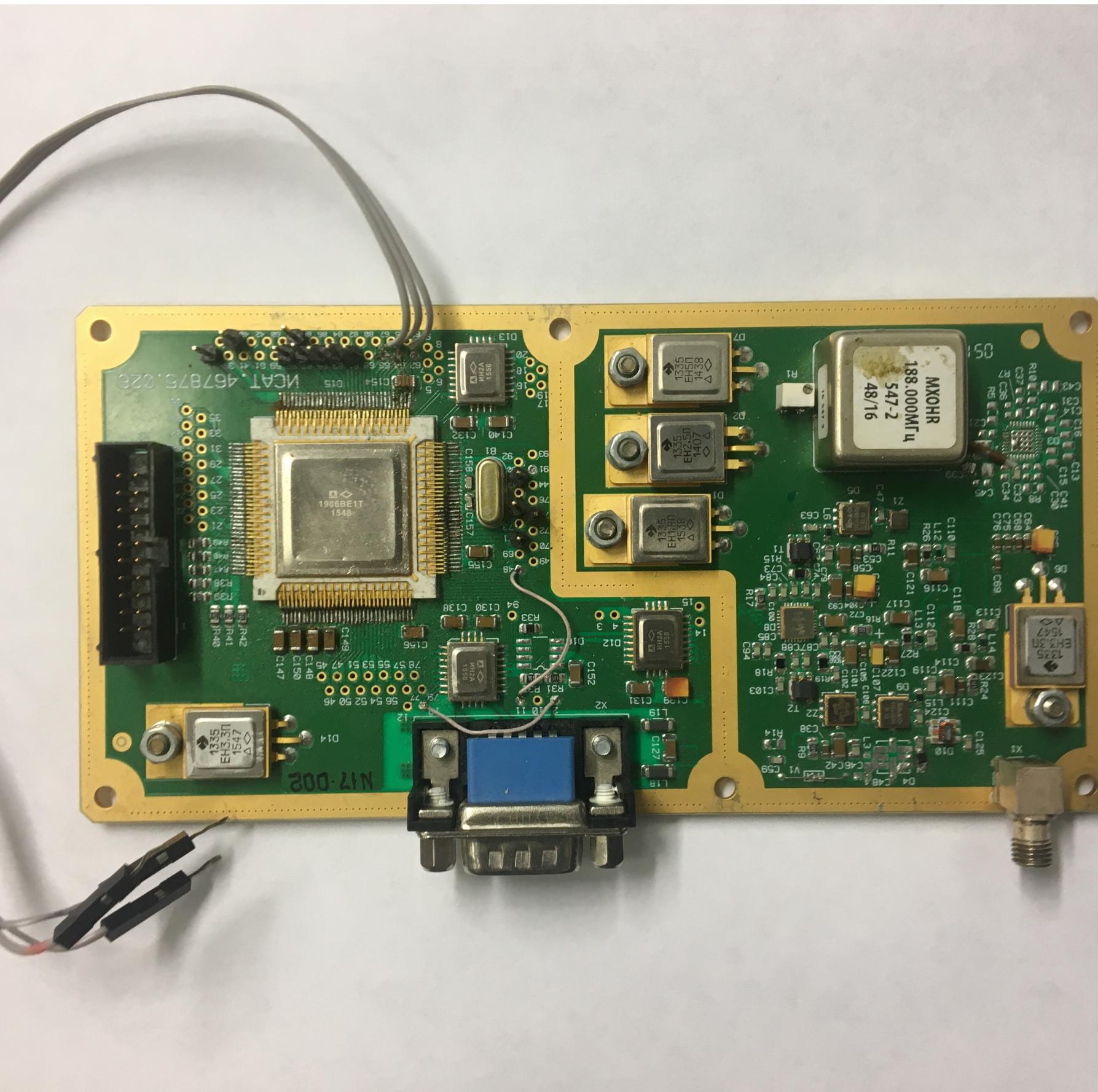


Чтение данных с com-порта и отображение данных в реальном времени

Елисеева Ульяна

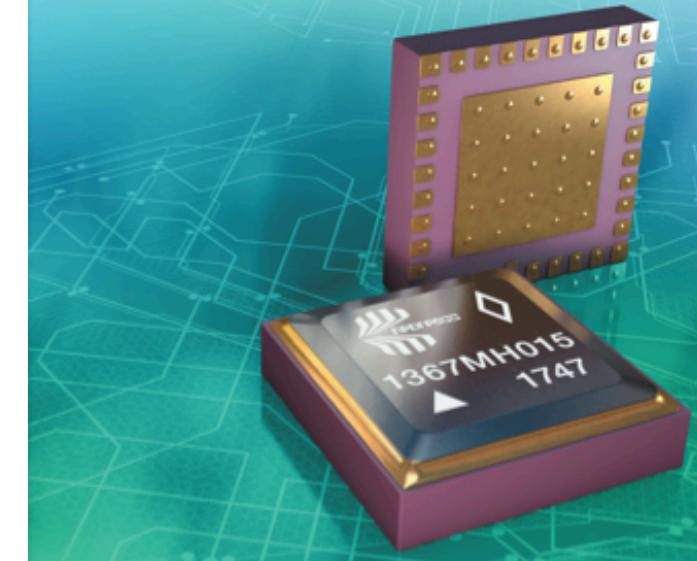
Синтезатор частот 1367МН015

Кремний-германиевый синтезатор частот на основе прямого цифрового преобразования предназначен для получения стабильного высокочастотного сигнала диапазона 600 МГц.



1367МН015

СБИС синтезатора частот на основе прямого цифрового преобразования



www.mri-progress.ru

Технические условия
АЕНВ.431320.051 ТУ
Децимальный номер КД:
ИЛТА.431322.004

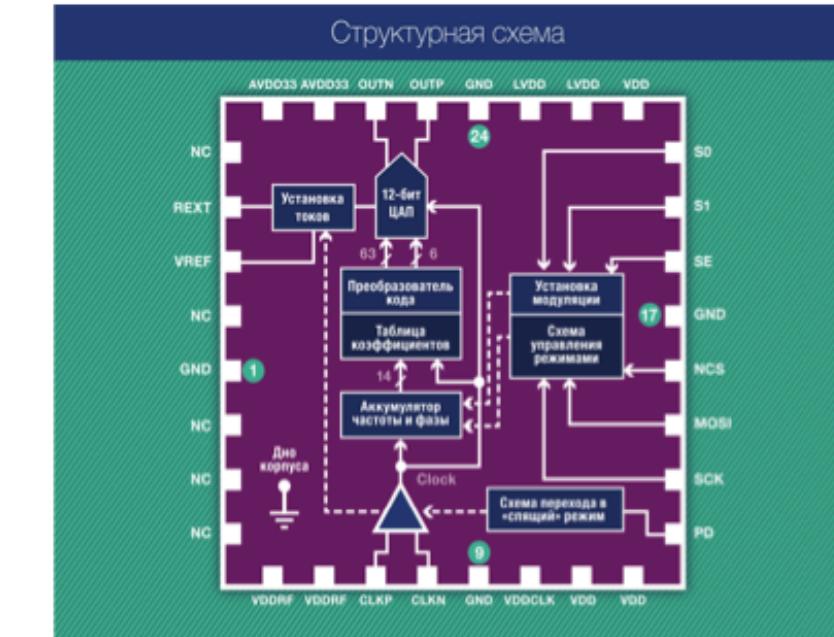
Применение микросхемы

- Радиолокация, РЭБ;
- Системы госопознавания, навигация;
- Связь наземная, авиационная, спутниковая;
- Широкополосная беспроводная связь;
- Измерительное оборудование.

Функциональный аналог AD9912 (Analog Devices Inc.).

Краткое описание микросхемы

Кремний-германиевая БИКМОП СБИС синтезатора частот на основе прямого цифрового преобразования предназначена для получения стабильного высокочастотного сигнала диапазона 600 МГц со сверхнизким шагом перестройки частоты, низким уровнем фазовых шумов и быстрой перестройкой частот, а также возможностью осуществления частотной, фазовой и ЛЧМ модуляции.



Электрические параметры

при при $V_{cc}=3,3$ В, $T=27^{\circ}\text{C}$

Параметр, единица измерения	Мин.	Тип.	Макс.
Ток потребления, мА	380	400	450
Частота тактового сигнала, ГГц	1		1200
Максимальная частота выходного сигнала, МГц			600
Входной ток ЦАП, мА	5		25
Интегральная линейность ЦАП, МЗР	0,5	1	2
Дифференциальная нелинейность ЦАП, МЗР	0,5	1	1,5

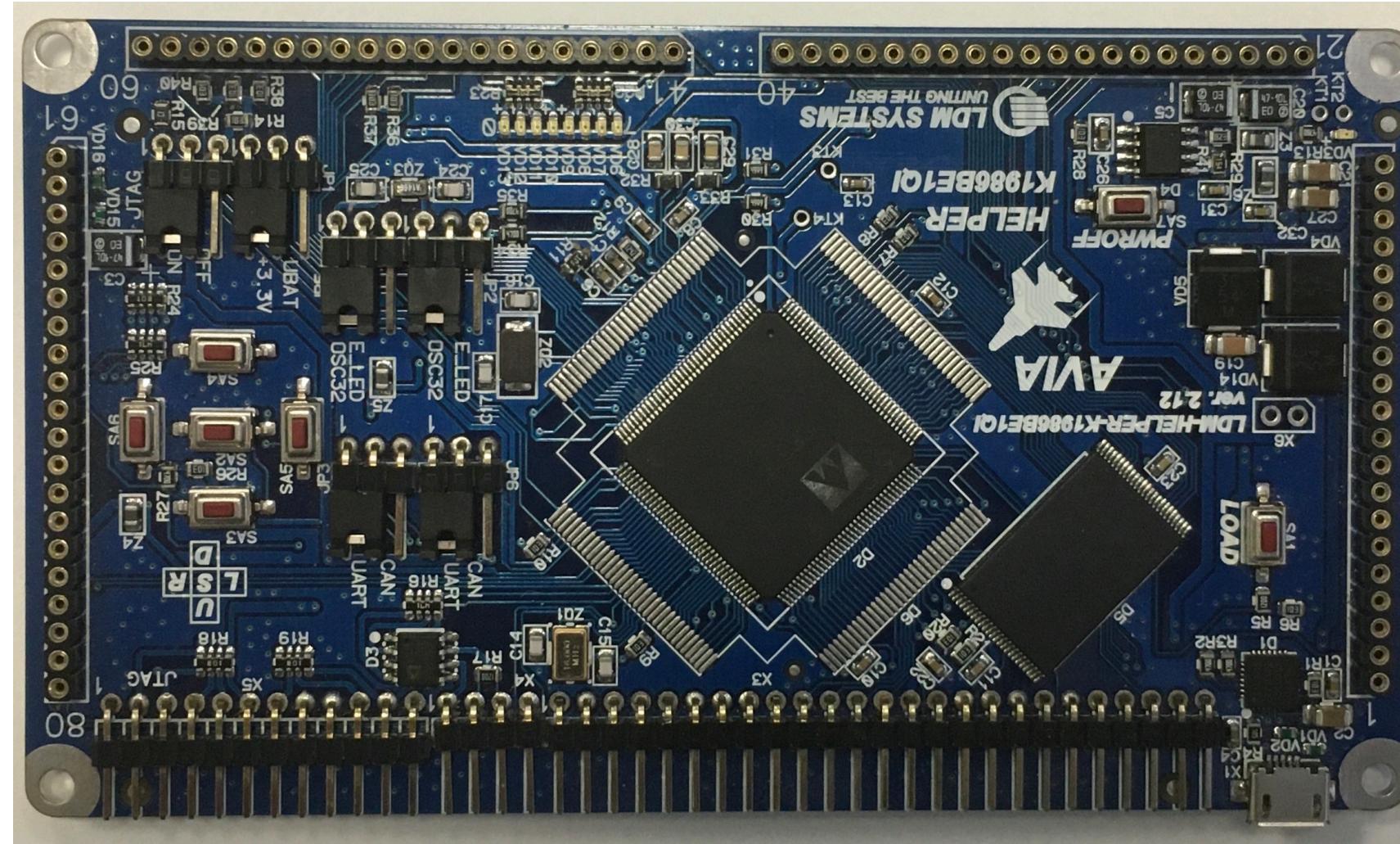
По вопросам приобретения:

Назначение выводов

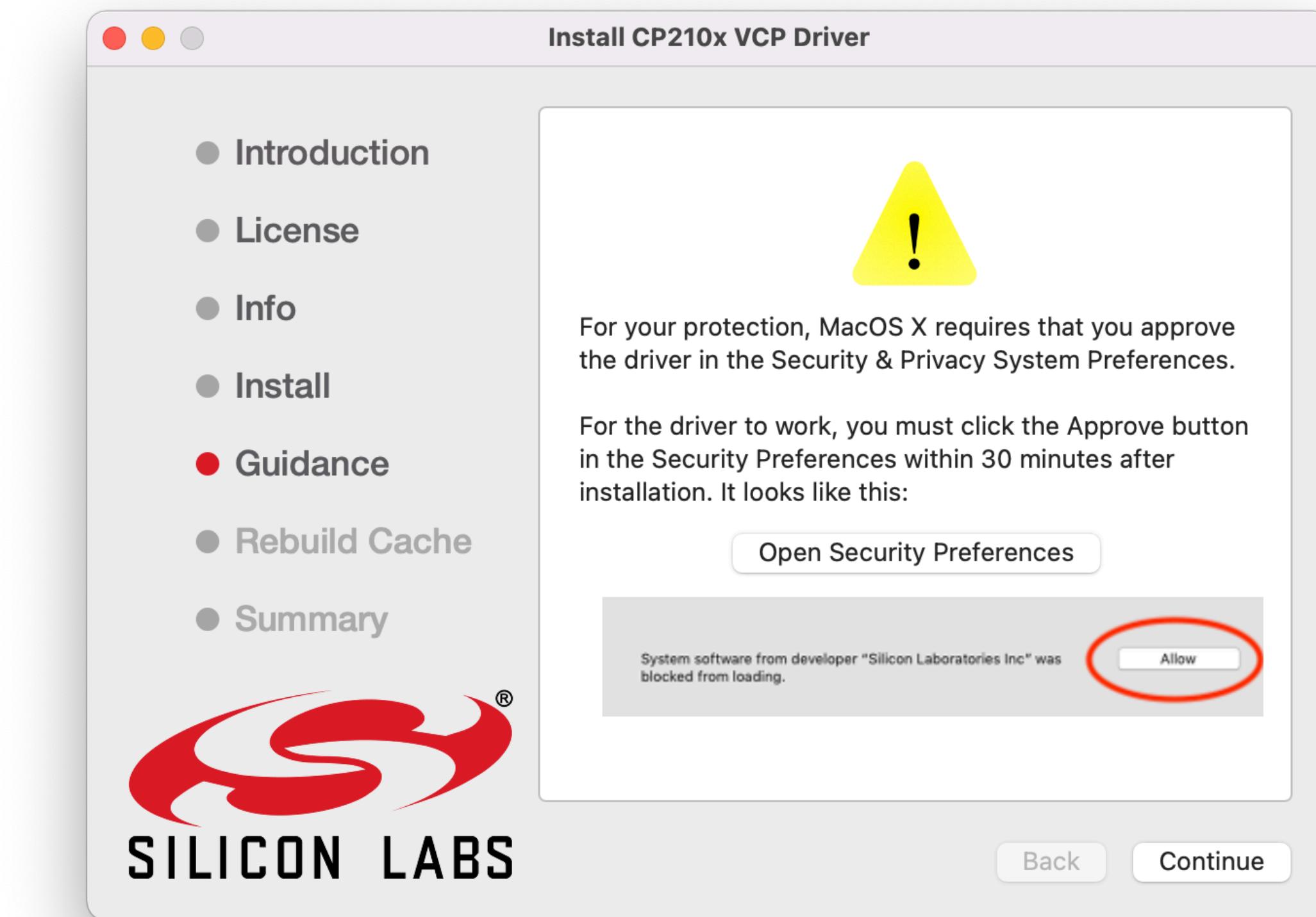
5,6,10-12, 21-23	Питание 1,8 В	2-4, 29, 32	Нет соединения
27, 28	Питание 3,3 В	7, 8	Вход опорного сигнала
25, 26	Выход ЦАП	30	Внешний резистор
13	Установка спящего режима	14-16, 18-20	Управление
1, 9, 17, 24	Земля (дно корпуса)	31	Внешнее опорное напряжение 1,2 В

1367MH015

Отладочная плата

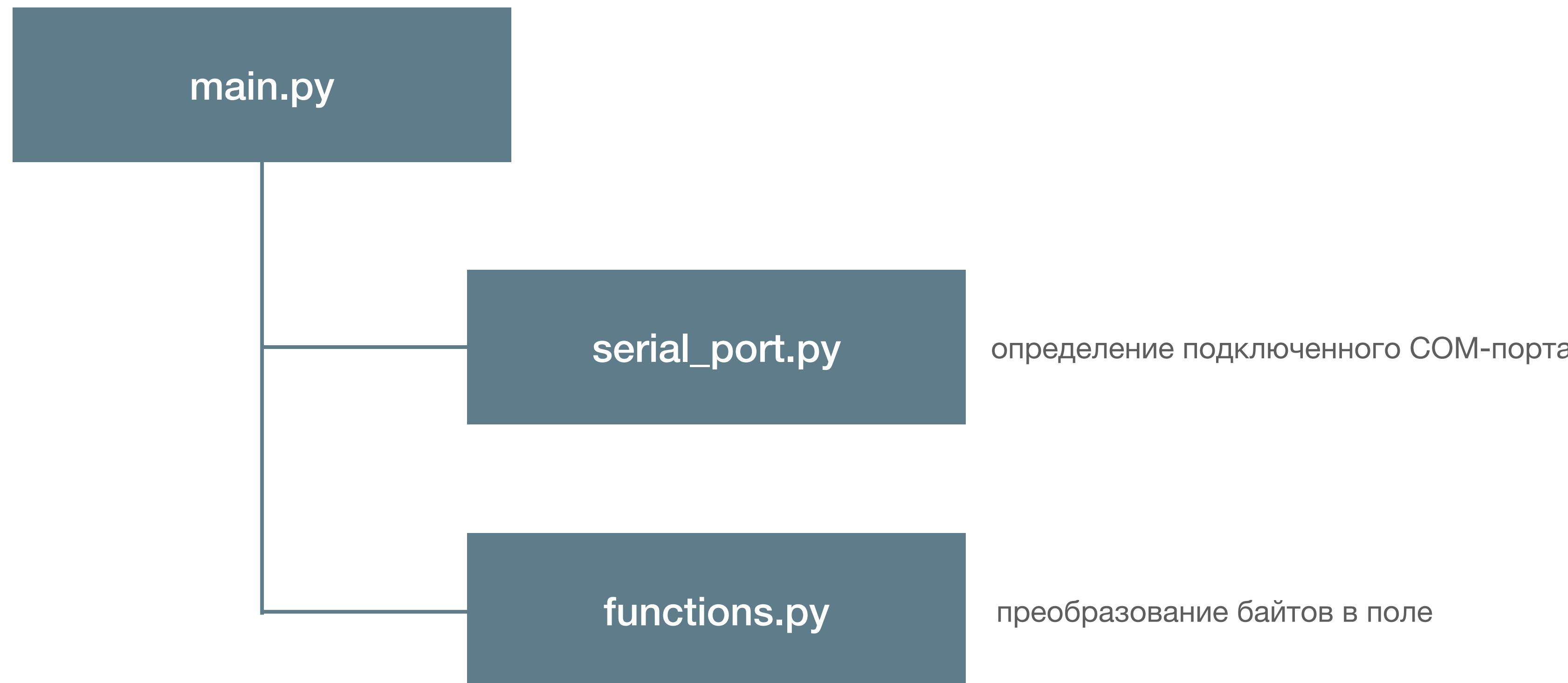


Отладочная плата для микроконтроллера 1986ВЕ1Т
для проверки работоспособности кода и поиска ошибок
на этапе макетирования



CP210x USB to UART Bridge VCP Drivers для подключения
микроконтроллера с последовательным интерфейсом UART через
(виртуальный СОМ-порт)

Структура



Импортируемые библиотеки

```
1 import collections
2 import serial
3 from matplotlib import pyplot as plt
4
5 from functions import calc_frequency
6 from serial_port import serial_port
```

serial_port.py

```
1 import serial
2
3 def serial_port():
4     """ Определяет номер подключенного com-порта Windows"""
5     ports = ['COM%s' % i for i in range(256)]
6     for port in ports:
7         try:
8             s = serial.Serial(port)
9             s.close()
10            return port
11        except serial.SerialException:
12            pass
```

Формат данных

```
FFCF2DCD2CE1D5FFC02DCD2CE1C6FFC1B5A12CE123FFC2B5A12CE124FFC33D762CE182FFC43D762C  
E1EAFFC3BDBE29E147FFC4BDBE29E148FFC5459329E1A6FFC6459329E1A7FFC7CD6729E104FFC8CD  
DB26E16CFFC74DB026E1CAFFC84DB026E1CBFFC9D58426E128FFCAD58426E129FFCB5D5926E187FF  
CA55CD23E1EFFFCBDDA123E14CFFCCDDA123E14DFFCD657623E1ABFFCE657623E1ACFFCFED4A23E1  
70FFCEE5BE20E171FFCF6D9320E1CFFFC06D9320E1C0FFC1F56720E11DFFC2F56720E11EFFC37D3C  
1DE1E3FFC275B01DE1E4FFC3FD841DE141FFC4FD841DE142FFC585591DE1A0FFC685591DE1A1FFC7  
05A21AE166FFC605A21AE167FFC78D761AE1C4FFC88D761AE1C5FFC9154B1AE123FFCA154B1AE124  
FFC9959317E1E8FFCA959317E1E9FFCB1D6817E147FFCC1D6817E148FFCDA53C17E1A5FFCEA53C17  
E10DFFCD258514E16BFFCE258514E16CFFCFAD5914E1C9FFC0AD5914E1BAFFC1352E14E118FFC235  
A211E180FFC1B57611E1DDFFC2B57611E1DEFFC33D4B11E13CFFC43D4B11E13DFFC5C51F11E19AFF  
C4BD930EE102FFC545680EE160FFC645680EE161FFC7CD3C0EE1BEFFC8CD3C0EE1BFFFC955110EE1  
84FFC84D850BE185FFC9D5590BE1E2FFCAD5590BE1E3FFCB5D2E0BE141FFCC5D2E0BE142FFCDE502  
08E106FFCCDD7608E107FFCD654B08E165FFCE654B08E166FFCFED1F08E1C3FFC0ED1F08E1B4FFC1  
6D6805E189FFC06D6805E17AFFC1F53C05E1D7FFC2F53C05E1D8FFC37D1105E136FFC47D1105E137  
FFC3FD5902E1FBFFC4FD5902E1FCFFC5852E02E15AFFC6852E02E15BFFC70D0302E1B9FFC80D0302  
E020FFC78D4BFFE07DFFC88D4BFFE07EFFC91520FFE0DCFFCA1520FFE0DDFFCB9DF4FEE039FFCC9D  
68FCE0A2FFCB1D3DFCE000FFCC1D3DFCE001FFCDA511FCE05EFFCEA511FCE05FFF2DE6FBE0BCFF  
CE255AF9E025FFCFAD2EF9E082FFC0AD2EF9E073FFC13503F9E0D1FFC23503F9E0D2FFC3BDD7F8E0
```

4 инф. байта

FF CF 2DCD2CE1 D5

7 байт

functions.py

```
def reverse_bytes(value):
    """Разворот байтов в обратном порядке"""
    pass
```

2D CD 2C E1 → E1 2C CD 2D

```
def sum_bytes(value):
    """Проверка пакета байтов по контрольной сумме"""
    pass
```

FF CF 2D CD 2C E1 D5

```
def calc_frequency(value):
    """Преобразование байтов в частоту"""
    pass
```

2D CD 2C E1 → 481.11212536459794 МГц

```
def calc_field(value):
    """Преобразование кода в поле"""
    pass
```

2D CD 2C E1 → 57407.51284123898 нТл

main.py



```
1 x = collections.deque([0]*maxsize, maxlen=15000)
2 y = collections.deque([0]*maxsize, maxlen=15000)
3
4 with serial.Serial(serial_port, baudrate=115200) as ser:
5     while True:
6         ser.read_until(expected=b'xff')
7         data = ser.read(6)
8
9         if sum_bytes(data):
10             data = calc_freq(data.hex()[2:10])
11
12             x.append(numbers)
13             y.append(data)
14             numbers += 1
15             iter += 1
16             if iter > 15:
17                 plt.plot(x, y)
18                 i = 0
```

