Tode Firmware

Firmware Development Guide

For Custom Engineering (Not a User Guide)

by TGit-Tech [http://www.tgit-tech.com]
Build Version: 21C9 / Last Updated: 2021-12-09

Table of Contents

1. Introduction

The Tode System

- Tode-RC = Handheld Remote Control Models
  - Model #AMP Arduino Mega Pro (No RF Module)
  - Model #AMPE32T30 Arduino Mega Pro + Ebyte E32-433T30D (1W/30dbm) RF module
  - Model #AMPE32T20 Arduino Mega Pro + Ebyte E32-433T20D (250mW/20dbm) RF module
  - Model #AMPXBEE Arduino Mega Pro + Digi XBee RF Module

- Tode-SideIO = Input/Output Stations
  - Model #TSIOST Tode SideIO with Screw Terminals
  - Model #TSIOAP Tode SideIO with Aviation Plugs

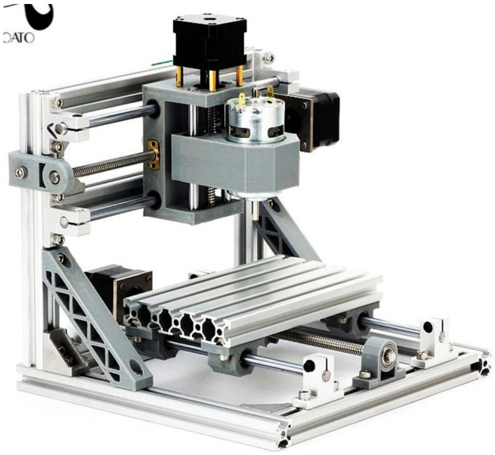
Manuals

- User Manual Operator Instructions including Setup and Wiring
- Hardware Development How to build the hardware including detailed circuit diagrams
- Firmware Development How to adjust and create firmware for the Tode

The Tode System is liscensed under the MIT Liscense. It's hosted on Github.com at:
https://github.com/TGit-Tech/Tode-RC

2. Workstation

2.1 Equipment

 ✔ Desktop Computer ○ Capable of running Arduino Sketch		

2.2 Software

Aduino Sketch	Doxygen			

2.3 Firmware Uploading

1.

3. Firmware

Each Item is

3.1 Overview

Items or List

3.2 Code Documents

Doxygen creates per release documentation to assist in recent changes.
This document is meant as a starting point.

3.3 Adding Device Driver

1. File **iDev.h**
   a) Assign a Device Type to the new Device

```
/*******************************************************************************//**
 * @defgroup DT [DT] Device-Type enumeration
 * @{
 **********************************************************************************/
// 0x[7] Device Types
#define DT_RW_ONOFF   0x7E    ///< On/Off Switching Device
#define DT_RO_ONOFF   0x7D    ///< On/Off Monitoring Device
#define DT_RO_PRESS   0x7C    ///< Pressure Device
#define DT_RO_TEMP    0x7B    ///< Temperature Device
#define DT_RO_DIST    0x7A    ///< Distance Sensing Device
#define DT_RW_STSTP3W 0x79    ///< Start Stop 3-Wire
///@}
```

   b) Prototype a Class for the Device Type

```
/***************************************************************************************************************//**
 * @class   OnOff
 * @brief   **DEVICE** Digital On/Off *Value* Control and Local Settings; inherits MenuValue.
 ******************************************************************************************************************/
class OnOff : public Device {
  public:
    OnOff(byte _TodeIndex, byte _RFID, HdwSelect* _Hardware=0);

    virtual int     IOValue() override;                                            // Stored on Tode
    virtual void    IOValue(int _Value) override;

};
```

2. File **iDev.cpp**
   a) CONSTRUCTOR - The Device Constructor
   - if (IsLocal) then create a SubList of Device Settings
     - Pin Assignment and Settings in “SubList = new MenuList(“Dev Setup”)
     - Store each Setting as MenuItem as below
     - PinX = SubList->Add(new PinSelect(“Pin X”,_RFID,_Hardware)

```
OnOff::OnOff(byte _TodeIndex, byte _RFID, HdwSelect* _Hardware):
  Device(_TodeIndex, _RFID)  {                DBINITAAL(("OnOff::OnOff(TodeIndex,RFID,Hardware)"),(_TodeIndex),(_RFID))
    ValueRange(0,1);
    SetNumberName(0, "Off");
    SetNumberName(1, "On");
    if (IsLocal) {
      if ( _Hardware==0 ) { DBERRORL(("OnOff::OnOff IsLocal but Hardware==0")) return; }
      SubList = new MenuList("OnOff Setup");
      IOPin = SubList->Add(new PinSelect("Pin",_RFID,_Hardware));
    }
}
```

   b) GET - Write the method required to READ the IOValue()
   - Retrieve the Device Settings in step #a above by PinX->Value()

```
int OnOff::IOValue() {                                                  DBENTERL(("OnOff::IOValue(GET)"))
  if ( IOPin == 0 ) { DBERRORL(("OnOff::Value GET - IOPin==0")) return -1; }
  if ( IOPin->Value() < 0 || IOPin->Value() > 70 ) { DBERRORL(("OnOff::Value GET - IOPin OUT OF BOUNDS")) return -1; }
  DBINFOAL(("OnOff::IOValue(GET) digitalRead"),(IOPin->Value()))
  return digitalRead(IOPin->Value());
}
```

   c) SET - Write the method required to WRITE the IOValue or make an empty one for Read-Only

```
void OnOff::IOValue(int _Value) {                         DBENTERAL(("OnOff::IOValue(SET): "),(_Value))
```

```
  if ( IOPin == 0 ) { DBERRORL(("OnOff::Value SET - IOPin==0")) return -1; }
  if ( IOPin->Value() < 0 || IOPin->Value() > 70 ) { DBERRORL(("OnOff::Value SET - IOPin OUT OF BOUNDS")) return -1; }
  DBINFOAAL(("OnOff::IOValue(SET) digitalWrite"),(IOPin->Value()),(_Value))
  digitalWrite(IOPin->Value(),_Value);
}
```

3. In **Sys.cpp** method Sys::BuildSetupMenu()
    a) Under comment // 5. Add Device
    - Add a new MenuName with the Text Name of the Device Type and Assigned Device TYpe in #1.a
    - Example 'AddDeviceList->Add(new MenuName("Distance", DT_RO_DIST));'

```
  // 5. Add Device
  //SetupMenu->Add(new MenuName("Devices", DeviceList = new MenuList("Devices")));                // 2. Devices
  SetupMenu->Add(new MenuName("Add Device", AddDeviceList = new MenuList("Add Device")));      // 2.1 Add Device
  AddDeviceList->Add(new MenuName("OnOff",     DT_RW_ONOFF));                                  //  2.1.1 OnOff
  AddDeviceList->Add(new MenuName("Pressure",  DT_RO_PRESS));                                  //  2.1.2 Pressure
  AddDeviceList->Add(new MenuName("Temp",      DT_RO_TEMP));                                   //  2.1.3 Temp
  AddDeviceList->Add(new MenuName("Distance",  DT_RO_DIST));                                   //  2.1.4 Distance
  AddDeviceList->Add(new MenuName("STSTP3W",   DT_RW_STSTP3W));                                //  2.1.5 Distance
```

    b) In method Sys::Loop()
    - Adjust Boundaires Check
    - if (0x7A <= _FinalKey && _FinalKey <= 0x7E)

4. In **LHdw.cpp** @ Tode::AddDevice
    a) Adjust Boundary Check in Tode::NewDevice
    - if (0x7F < _DTKey || _DTKey < 0x79) { DBERRORL(("Tode::NewDevice. DTKey OUT OF BOUNDS")) return 0; }

    b) Adjust Boundary Check in Tode::AddDevice
    -

    c) fill in the appropriate this->Add() if else line for the new Device Type
    - } else if (_DTKey == DT_RW_STSTP3W) {
    -    Devices[_RFID] = this->Add(new STSTP3W(TodeIndex,_RFID,Hardware));
    -    return Devices[_RFID];
    -   }

*Note after we add device the display goes off

5. To allow the new device recognition on Configuration
    a) Received Configuration Data goes to Method
    - RF->Packet->SaveTodeConfig
        - void RxPacket::SaveTodeConfig(int _EEAddress)
        - No Checks just writes EEPROM bytes
    - TargetTode->EELoadDevices

    b) So – We need to make sure EEPROM loading EELoadDevices is setup correctly
    - Which if it's not 0xFF it goes to AddDevice()
    - Which adds the device exactly as-if it was added by Menu?

1. Add another Class for the Device in -

4. *Class*

4.1 E32

The E32 Class if for EByte E32 Radio Modules. It requires the RFC class for sending TxPacket and RxPacket.

4.2 RFC – RF Communications

RFC handles the RF Protocol and should be Radio Module in-sensitive.

RFC stores the first 58-bytes of packets in an array and any bytes above 58 are stored in a Queue.

The first 12-bytes in the 58-byte array are assigned by @defgroup PKB Common Packet Byte RFIDes
[0][1] = TO RF Address 0xFFFF form
[2] = Channel
[3] = Security HighByte
[4] = Security LowByte
[5] = Packet Type Determined by PKT – Radio Packet Types
[6][7] = FROM RF Address 0xFFFF
[8] = Tode Version Keeps track of Configuration changes for updates

Every Packet contains --

Class RadioI (Interface) in RFC is used to Interface various Radio (E32) modules
Object 'RF' is an instance of a RadioI implementation.

4.2.1 Rx versus Tx

RxPacket::RxByte to add each received byte
TxPacket::TxByte to add each transmission byte