Documentation for the 1-Wire Net Public Domain Kit

Version 2.00

Steps to Use the 1-Wire Public Domain Kit

Step 1: ***Do we support your platform?***

There are two sets of portable source files. The first set is general purpose and is intended for platforms that already have the primitive link-level 1-Wire Net communication functions. This is the lowest level that is hardware dependent and is called ‘general’.

The ‘userial’ set of portable source files assumes that the user has a serial port (RS232) and wishes to utilize our 'Universal Serial 1-Wire Line Driver Master chip' called the DS2480. This chip receives commands over the serial port, performs 1-Wire Net operations and then sends the results back to the serial port.

Platforms currently supported:

Operating System	Compiler	Library	Name
Windows 32-bit	Visual C	userial	uWin32VC
Windows 32-bit	GNU	userial	uWin32GNU
Windows 32-bit	TMEX	general	gTMEXVC
Linux	GNU	userial	uLinuxGNU
Beos	GNU	userial	uBeosGNU

Yes:

Use the existing link files. You may also want to download example builds from the Public Domain kit (\’uWin32VC’, \’gTMEXVC’, \’uWin32GNU’, \’uLinuxGNU’, \’uBeosGNU’)depending on the platform you’re using.

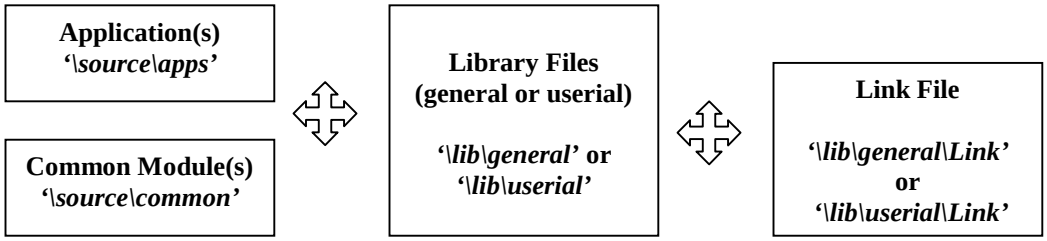
No:

Look at other links as examples. We recommend looking at ‘lib\general\Link’ or ‘lib\userial\Link’ section in the Public Domain Kit. Please refer to pages 8-11 of this document for General and Userial Application Interface Code.

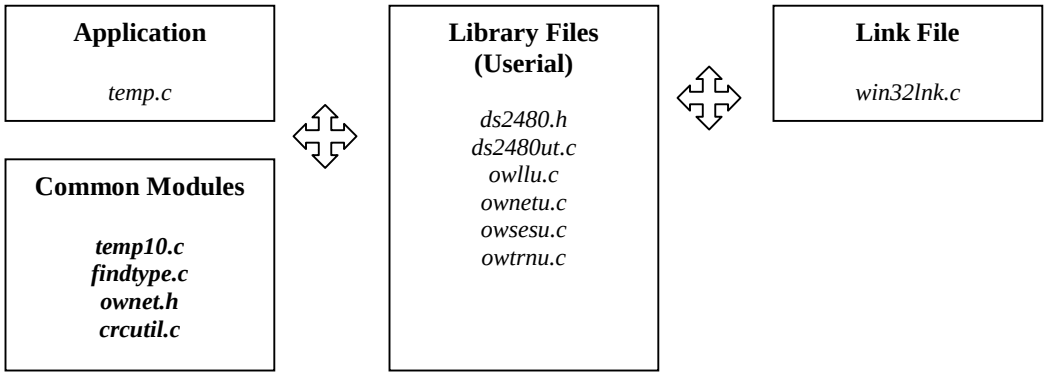
Serial port? ‘userial’
else use ‘general’

Step 2: ***How do you build applications?***

You will need files from each of these directories:



Example: '\source\apps\temp'



Device Listing

Example applications and 'C' modules for specific 1-Wire devices are listed below by Dallas Semiconductor part number.
This section is provided to quickly find code resources specific to a 1-Wire device type.

- **DS1820/DS18S20/DS1920**
  Example Code for the DS1820/DS18S20/DS1920 can be found in:
  '\source\common\'
  temp10.c
  An example application can be found in:
  '\source\apps\temp'
  '\source\apps\mweather'
  temp.c
  mweather.c

- **DS1921**
  Example Code for the DS1921 can be found in:
  '\source\common\'
  thermo21.c
  thermo21.h
  An example application can be found in:
  '\source\apps\thermo'
  thermodl.c
  thermoms.c

- **DS1963S**
  Example Code for the DS1963S can be found in:
  '\source\common\'
  ibsha18.c
  ibsha18o.c
  ibhsa18.h
  An example application can be found in:
  '\source\apps\ibsha'
  initcopr.c
  initrov.c
  debit.c
  isbhaut.c

- **DS2405**
  Example Code for the DS2405 can be found in:
  ‘\source\common\’
  swt05.c
  An example application can be found in:
  ‘\source\apps\swtsngl’
  swtsngl.c

- **DS2406/DS2407**
  Example Code for the DS2406/DS2407 can be found in:
  ‘\source\common\’
  swt12.c
  An example application can be found in:
  ‘\source\apps\swtlpop’
  ‘\source\apps\mweather’
  swtoper.c
  swtloop.c
  mweather.c

- **DS2409**
  Example Code for the DS2409 can be found in:
  ‘\source\common\’
  swt1f.c
  An example application can be found in:
  ‘\source\apps\coupler’
  coupler.c
- **DS2423**
  Example Code for the DS2423 can be found in:
  ‘\source\common\’
  cnt1d.c
  An example application can be found in:
  ‘\source\apps\counter’
  ‘\source\apps\mweather’
  counter.c
  mweather.c

- **DS2450**
  Example Code for the DS2450 can be found in:
  ‘\source\common\’
  atod20.c
  An example application can be found in:
  ‘\source\apps\atodtst’
  ‘\source\apps\mweather’
  atodtst.c
  mweather.c

1-Wire Application Interface

This section provides a brief description of the API functions contained in the 'general' and 'userial' 1-Wire libraries.

Link-Level 1-Wire Net functions:

owTouchReset

Reset all devices on the 1-Wire Net. Result of function indicates if any devices were detected

owTouchBit

Send and receive 1 bit from the 1-Wire Net

owTouchByte

Send and receive 8 bits from the 1-Wire Net

owWriteByte

Send 8 bits to the 1-Wire Net and verify the echo received matches.
(constructed from owTouchByte)

owReadByte

Receive 8 bits from the 1-Wire Net by sending all 1's (0xFF) and letting the slave change the echo.
(constructed from owTouchByte)

owSpeed

Set the communication speed of the 1-Wire Net to Normal (16K bits) or Overdrive (142K bits).
All 1-Wire devices at least support the Normal communication rate.

owLevel

Set the 1-Wire Net line level to Normal (5V weak pullup), Power Delivery (5V strong pullup), or Program Level (12V EPROM programming level).
Power delivery only required by some 1-Wire devices.
Programming level only required to write EPROM based memory 1-Wire devices.

owProgramPulse

Timed programming pulse for EPROM 1-Wire device writing.
Can be constructed from owLevel.
Only required to write EPROM based memory 1-Wire devices.

Network-level 1-Wire Net functions:

owFirst

Search to find the 'first' 1-Wire device on the 1-Wire Net.
All 1-Wire Net devices have a unique 64-bit serial number.
The order the devices are found is serial number dependent.
The serial number found can be retrieved after this function using owSerialNum.

owNext

Search to find the 'next' 1-Wire device on the 1-Wire Net based on the last search done. The serial number found can be retrieved after this function using owSerialNum. If owNext returns FALSE then the end of the search has been found. Calling owNext again will reset the search and find the 'first' device again.

owSerialNum

Retrieve or set the currently selected device serial number.

owFamilySearchSetup

Setup the following search (owNext) to find a specific family type.
The first 8 bits of the unique serial number indicate the 'family' that the device belongs to. The 'family' lets the application know what type of commands the device requires. The owSerialNum function must be called after the search has been performed to verify the correct family type was found. If it is not the correct family type then there are no devices of that type on the 1-Wire Net.

owSkipFamily

Skip all of the family type that was found in the last search. The next search (owNext) will find a new type or come to the end of the search.

owAccess

Select the current device by Serial Number. The selection is done by resetting the 1-Wire Net, sending the 'MATCH ROM' command followed by the current serial number. At the end of this operation only the current device is listening on the 1-Wire Net for a device specific command.

owVerify

Selects and verifies that the current device by Serial Number is on the 1-Wire Net. This function uses the ‘search' command to select and verify the device is in contact with the Net.

owOverdriveAccess

Select the current device by Serial Number and place it and the 1-Wire Net into Overdrive communication speed.

Transport-level 1-Wire Net functions:

owBlock

Send and receive blocks of data to the 1-Wire Net. A reset is optionally done on the 1-Wire before the data is sent. This API is more efficient than sending data with multiple 'owTouchByte' calls.

owReadPacketStd

Read standard UDP packet structure from a memory 1-Wire device. See 'Dallas Semiconductor Application Note 114' for a description of this structure.
(function options may change on future versions of this code)

owWritePacketStd

Write standard UDP packet structure into the memory of 1-Wire device.
(function options may change on future versions of this code)

owProgramByte

Program a byte to an EPROM based 1-Wire device memory.
(function options may change on future versions of this code)

File-level 1-Wire Net functions:

owReadFile

Read a TMEX file structure file from the memory of a 1-Wire device.
Not all 1-Wire devices supported.
See 'Dallas Semiconductor Application Note 114' for a description of the TMEX file structure.
(function options may change on future versions of this code)

owFormatWriteFile

Format and then write a TMEX file structure file into the memory of a 1-Wire device. Not all 1-Wire devices supported. See 'Dallas Semiconductor Application Note 114' for a description of the TMEX file structure. (function options may change on future versions of this code)

Session-Level 1-Wire Net functions:

owAcquire

Attempts to acquire a 1-Wire net

owRelease

Releases the previously acquired a 1-Wire net

General API Code

The 'general' code set sends commands that rely on the following 1-Wire Net link-level functions.

The following is a description of the API needed to port this code set to any platform. See the 'TODO.C' and 'TODOSES.C' files in the directory '\source\lib\general'.

Required API that must be implemented for the 'general' code set to function:

owTouchReset
Reset all devices on the 1-Wire Net. Result of function indicates if any devices were detected.

owTouchBit
Send and receive 1 bit from the 1-Wire Net

API that can be generated from the required API:

owTouchByte
Send and receive 8 bits from the 1-Wire Net. This can be contructed from 8 calls to owTouchBit but it may be for efficient to create this API.

owWriteByte
Send 8 bits to the 1-Wire Net and verify the echo received matches.
(constructed from owTouchByte)

owReadByte
Receive 8 bits from the 1-Wire Net by sending all 1's (0xFF) and letting the slave change the echo.
(constructed from owTouchByte)

Optional API that may be needed for the platform or for a specific application.

owSpeed

Set the communication speed for the 1-Wire Net to Normal (16K bits) or Overdrive (142K bits). All 1-Wire devices at least support the Normal communication rate. This API need only be implemented if Overdrive communication rate is desired.

owLevel

Set the 1-Wire Net line level to Normal (5V weak pullup), Power Delivery (5V strong pullup), or Program Level (12V EPROM programming level). Power delivery only required by some 1-Wire devices such as the DS1820, DS2450, and DS1954. Programming level only required to write EPROM based memory 1-Wire devices.

owProgramPulse

Timed programming pulse for EPROM 1-Wire device writing. Can be constructed from owLevel. Only required to write EPROM based memory 1-Wire devices.

msGettick

Return an increment millisecond counter. This is used in several of the sample applications.

msDelay

Delay at least the specified number of milliseconds. This is used in several of the sample applications.

owAcquire

Attempts to acquire a 1-Wire net

owRelease

Releases the previously acquired a 1-Wire net

Userial API Code

The 'userial' code set sends commands designed for the DS2480 'Universal Serial 1-Wire Line Driver Master chip'. It can be made to work on any serial port that can do 9600, N, 8, 1.

The following is a description of the API needed to port this code set to any platform. See the 'TODO.C' file in the directory '\source\lib\userial'.

Required API that must be implemented for the 'userial' code set to function:

ReadCOM

Read a specified number of bytes from the serial COM port. This API must timeout if the specified number of bytes have not arrived. The timeout is dependent on the 1-Wire operations the platform but a good rule of thumb is 10 - 20ms per byte.

WriteCOM

Write a specified number of bytes to the serial COM port.

FlushCOM

Allow any pending write operations to complete. Clear any bytes read.

BreakCOM

Send a 'BREAK' on the serial COM port last at least 2 milliseconds.

msDelay

Delay at least the specified number of milliseconds. This is used in the DS2480 detect sequence.

Optional API that may be needed for the platform or for a specific application.

OpenCOM

Open the specified serial COM port for communication. Most High-level OS's will need this. If not needed then just return success. Start the COM port out at 9600, N, 8, 1.

CloseCOM

Close the previously opened (OpenCOM) serial COM port. Most High-level OS's will need this.

SetCOMBaud

Change the serial BAUD rate to the rate specified. This need only be supported if Overdrive communication speeds are desired.

msGettick

Return an increment millisecond counter. This is used in several of the sample applications.