OpenPAYGO Link Manual

Overview

OpenPAYGO Link (OPL) is a communication protocol created as an open source standard solution for interfacing Solar Home Systems (SHS) with appliances and accessories. It is defined at hardware and software levels and features a half-duplex communication over a single wire with a multi-drop connection topology. It ensures reliable data transmission, even with different ground potentials between nodes, thanks to an off the shelf transceiver. Although it is not intended to transmit long frames it supports any variable-length payload. These characteristics make it a powerful and inexpensive alternative.

Hardware specifications

Device selection

As stated before OPL is based on the UART protocol configured to work in 9 bit address mode, thus it is required for the hardware to support it. The following table contains some of the most common targets that support this feature with some additional comments.

Table 2. MCU families supporting UART in 9-bit address mode

Target	Comments	Reference
STM8	Supports 8-9 bit address mode with 4 bit addresses	*22.3.7 Multi-processor communication*
STM32	Supports 8-9 bit address mode with 4 bit addresses	*27.3.6 Multiprocessor communication*
PIC18	Supports only 9 bit address mode	*31.3 Asynchronous Address Mode*
ATmega382	Supports 8-9 bit address mode	*9.9 Multi-processor Communication Mode*
MSP430	Supports 9 bit address mode	*15.3.3.3 Address-Bit Multiprocessor Format & 15.3.2 Character Format*
Nordic (e.g. nRF24E1)	Supports 9 bit transmission	*0.9 Serial Interface & 10.9.5 Multiprocessor Communications*

NXP	Supports 9 bit address mode	*9.12 USART0/1*
Renesas (e.g. RX130)	-	*27.4 Multi-Processor Communications Function*
Maxim Integrated	Supports 9 bit address mode	*7.5.8 Multidrop Mode Support*
Espressif (ESP8266 & ESP32)	No hardware 9 bit transmission support, but it can be emulated by software	*9 bit Software Serial library*
Raspberry Pi	Supports 9 bit transmission	*Bit patch* / *tty driver*

In some devices, like a PC, depending on the hardware (usually a USB-UART adapter) and software, the 9 bit address feature can be emulated using mark & space parity. The PySerial library for Python supports mark & space parity (*tested and works*).

The firmware is designed to be lightweight, with a minimal RAM usage, so even some of the most constrained microcontrollers on the market should be able to run it.

LIN transceiver

On the electrical side, there is no requirement for logical voltages, it can be either 3.3V CMOS or 5V TTL depending on the target, as OPL is based on a standard LIN transceiver (only the hardware side) and a third common voltage is used on the communication line. The LIN transceivers on average accept up to 30V DC on the power and data lines, however it is recommended to use 12V DC in order to be compatible with the majority off-grid solar household appliances.

OPL is designed for relatively high DC current (5-10A per port) systems, where good and low-impedance conductors are not always guaranteed to be used, thus creating voltage differences between the SHS and appliance GNDs. This situation can affect the communication integrity, so to verify the reliability of the transmission method a high current test was performed at 2400 bauds using the MCP2003 LIN transceiver.

Supply Volt	Current	Core area	Cable length	ΔV in GND	Result
12.74	5.04	0.50	3.70	1.23	Ok
12.74	8.34	0.50	3.70	2.07	Ok
12.74	10.18	0.50	3.70	2.56	Ok
11.74	9.10	1.50	3.00	0.36	Ok (115200b)

Since the test the default speed has been increased, it must be repeated.

Firmware specifications

The firmware is written in C language, using standard and open source libraries and only requires the Hardware Abstraction Layer (HAL) for the target where it is going to be implemented. The needed functionalities are:

- 1 ms counter
- GPIO interface (direct access to the registers or through an API)
- UART interface (direct access to the registers or through an API)
- EEPROM or any other non volatile memory interface

If using an off the shelf HAL, please make sure that it supports UART in 9 bit addressing mode, or at least that it can be modified in order to do so. Also verify that the RX pin can be read as a GPIO input when configured as a UART, otherwise connect RX to another GPIO set as an input.

Network

OPL runs over 9 bit UART transmitted with a LIN transceiver in a multidrop (many nodes sharing the same bus) half-duplex configuration.

Each of the nodes has a unique address within the network and, although some devices provide hardware that can detect an address match, the hardware is only used to distinguish between data and address, which is a much more common feature. The address match detection itself is done in software as part of OPL, however it is tightly coupled with the physical layer as OPL needs to control in real time the received bytes to handle a new stream of data if it's coming or check if the previous one has been received completely.

Datagram

The OPL frame is composed of 5 main sections: break, sync, header, payload and footer:

- The break is necessary to wake up the LIN transceiver with transmissions speeds higher than 4800 bauds. Otherwise the first byte of each transmission might get corrupted.

- The sync byte (0x55) is used as a sacrifice byte to ensure that all the bytes of the header are received properly, in case that the break is not sufficient. It can be considered optional as it will be ignored by the ISR.

- The header is made up of 2 bytes. The first byte contains the source and destination address, so that it can be processed as fast as possible. As the OPL frame supports any value from 0 to 255 in the data section, a terminator character cannot be used to detect

the end of the frame as it might not be unique within it. To address this problem the length of the payload is explicitly indicated in the lowest 7 bits of the second header byte, allowing payloads to be up to 127 bytes in length. The most significant bit of the second byte contains the data/command flag (0 = data, 1 = cmd).

- As stated before, the payload supports any 8 bit value and can be as long as 127 bytes.
- The footer contains only a 16 bit CRC (CRC-16/CCITT-FALSE) represented in 2 bytes with network (big) endianness. The CRC is calculated for the header and the payload, including the src/dest byte (*The advantage is that we can detect false address positives*).

-	*-*	*0*		*1*		*Len*	*Len + 1*	*Len + 2*
Break	Sync	Src	Dest	D/C	Len	Payload	CRC (MSB)	CRC (LSB)

Operation

Addressing

When a slave is connected a handshake occurs and the master assigns an available local ID to the slave. The master always starts the communication, sends a request and expects an answer within a specified time.

However, depending on the application, the master might need to send or receive from a slave that has a Unique Identifier (UID) such an IPv6 address or a manufacturer serial number. The API provides a bridge which translates the OPL local network address to any UID. If a device is reconnected and gets a new local address, the translation list is updated, so that the upper layers can still send and request to the same UID.

The UID, if there is one, is requested during the handshake, otherwise a UID based on the local address is used.

OCF Resources

OPL has a small implementation of the OCF resource model, which uses CoAP and CBOR. In the current state of the development, each slave exposes a set of resources that can be read or modified through a RESTful API. The methods include GET & PUT as the devices application is really constrained and does not require creation or deletion of resources.