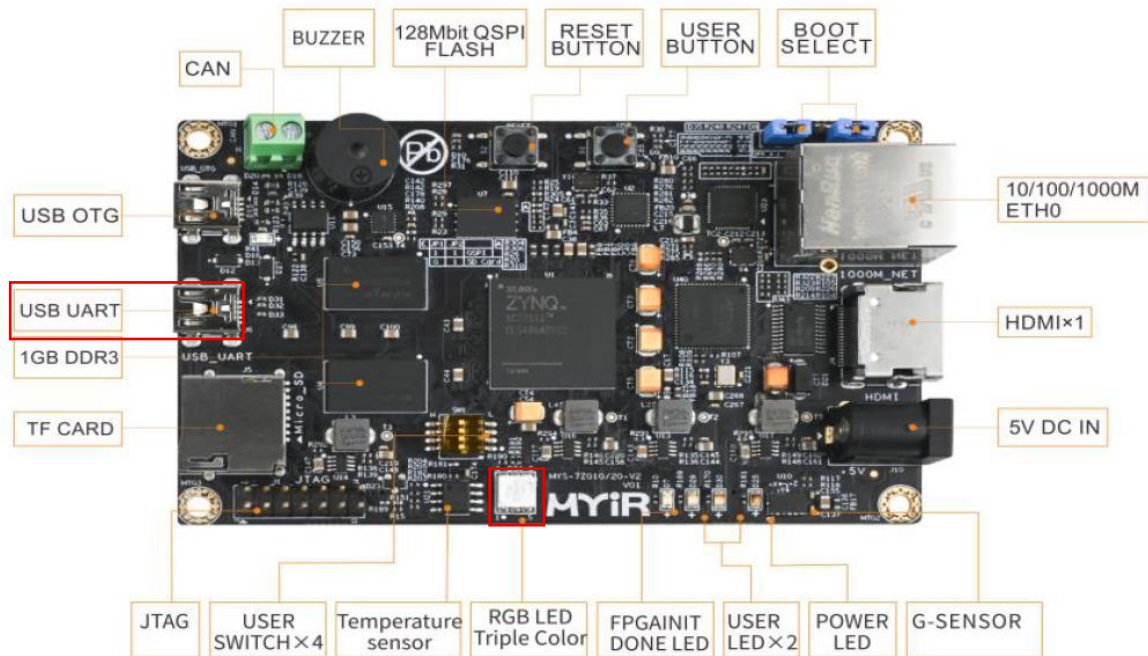


Getting started with OCB UT1 / ZTURN-V2 (v1.2-450)



- 1- Connect USB-UART to USB port on a PC. This will power on the board and you don't need to connect the external 5V PS.
- 2- Configure the USB port to be UART COM port with the following settings:
 - 115200 bit/s, 8-bits, no parity, 1 start, 1 stop
- 3- Connect a terminal on the configured COM port and reboot the board, the system will print the following and is listening for command interpreter client (CI) to connect on TCP port #50000 and DAQ client on TCP port #50001 (see blue arrow on next figure)

New features on v1.1-448 & higher (see orange arrow on next figure):

- **TFTP Server** working on partition '2:' = SD card ROOT partition
 - o By setting the remote file name beginning with \$DIR, you will get the list of files/directories in the required directory (e.g. \$DIR:2: will send a file containing the directory of 2: i.e. SD card ROOT partition)
 - o Tree separator is '/'
- **UDP Logger** pushing data on destination port #50005 and IP address set in 2:/CFG/LOGGER.INI file (see example on web site, ROOT directory)
 - o Data pushed even if no receiver is listening (UDP protocol)
 - o IP can be changed directly on the SD card or by TFTP with this file if CFG directory is created
 - o please create dir. and copy ROOT/CFG/LOGGER.INI to same tree in the SD card on 1st access
- `enablePrompt <En>` new command in command interpreter to enable/disable the prompt (enabled by default)

```

-----
FreeRTOS Unit Test 1 (Single SIM FEB, ping memory DMA transfer)
Version 1.1 (TFTP + UDP Logger)
FAT Partition '2:' mounted
COM TelnetServer CTOR:Creating 2 Clients Task
COM TelnetServer CTOR:Done
COM CTOR:Creating Main & Logger Task ...
COM MainTask CTOR:Creating Network Task ...
COM NetworkTask CTOR:Creating Emac Input Task ...
COM NetworkTask CTOR:Done
COM MainTask CTOR:Done
LOG CTOR
LOG CTOR:Done
COM CTOR:Done
MAIN:DAQ Sim mode
MAIN:Add commands to CI List with access
MAIN:Starting tasks scheduler...
MT:Main Task started
MT:Phy Reset...
MT:Lwip init...
MT:Starting Network Task...
NT:Network Task started
NT: MAC address : 8C-1F-64-D8-80-0A
WARNING: Not a Marvell or TI or Realtek or Xilinx PCS PMA Ethernet PHY or ADI Ethernet PHY.
Start PHY autonegotiation
Waiting for PHY to complete autonegotiation.
autonegotiation complete
Using default Speed from design
link speed for phy address 7: 1000
NT:Starting Emac Input Task...
EI:Emac Input Task started
NT:Starting DHCP client...
NT:Starting DHCP client loop...
MT:DHCP request success
Board IP : 10.195.49.227
Netmask : 255.255.248.0
Gateway : 10.195.48.1
LOG:Start
LOG:Started to 10.194.51.80, port #50005
TLC0:Resume Clients Task
TLC0:'DAQ' Telnet Client Task started
TLC0:Starting listening loop on port #50001
TLC1:'CI' Telnet Client Task started
TLC1:Starting listening loop on port #50000
TLC0:Resume Clients Task done
COM TFTP Server:Init
COM TFTP Server:Initialized
MT:System ready

```

MAC and IP addresses can be seen on the screen on blue arrows.

4- Connect the 2 clients, this will show:

```

TLC0:'DAQ' Telnet Client Task started
TLC0:Starting listening loop on port #50001
TLC1:'CI' Telnet Client Task started
TLC1:Starting listening loop on port #50000
TLC0:Resume Clients Task done
COM TFTP Server:Init
COM TFTP Server:Initialized
MT:System ready
TLC0:'DAQ' Client connected on 10.194.51.80, port #50001
TLC1:'CI' Client connected on 10.194.51.80, port #50000
TLC0:'DAQ' Client disconnected on port #50001
TLC1:'CI' Client disconnected on port #50000

```

5- Disconnect the 2 clients, this will show:

```

TLC0:'DAQ' Telnet Client Task started
TLC0:Starting listening loop on port #50001
TLC1:'CI' Telnet Client Task started
TLC1:Starting listening loop on port #50000
TLC0:Resume Clients Task done
COM TFTP Server:Init
COM TFTP Server:Initialized
MT:System ready
TLC0:'DAQ' Client connected on 10.194.51.80, port #50001
TLC1:'CI' Client connected on 10.194.51.80, port #50000
TLC0:'DAQ' Client disconnected on port #50001
TLC1:'CI' Client disconnected on port #50000

```

Warning: CI client disconnection is detected automatically by the OCB but DAQ client disconnection is detected only during a DAQ transmission, meaning an abnormal situation since disconnection should be done when the readout is disabled.

To disconnect properly for DAQ and CI, one must:


- a- Send the CI command `COM_DaqDisconnect` to force a disconnection

b- Disconnect the CI client

If not done, the DAQ client will remain connected and a new one couldn't be connected.

6- CI Commands with 'help':

```
help
help      : Display this help
hint      : Display command list
quit      : Quit and disconnect command interpreter
enablePrompt : <En>: Enable Prompt En=(0,1)
HW_FwVersion : Get Firmware Version
UT1_EnTHGLG : <T> <HG> <LG>: Enable Sim FEB with Time, HG, LG (0,1)
UT1_SetParam : <CH/GTS> <GTS/EVT> <#Gate> <#EmptyGts> <ChValSpan>: Set Sim FEB with #CH/GTS(1,64) #GTS/EVENT(1,4), Gate Number(0,65535), #empty GTS(0,10) and CH
value span GTS(1,16)
UT1_ResetCnt : Reset Sim FEB GTS & OCB Event counters
UT1_EnReadout : <En>: Enable Readout En=(0,1)
UT1_SetDaqDelay : <Delay>: Set Daq delay per event in ms (1,1000)
COM_DaqDisconnect : Disconnect DAQ
[OK]
UT1>enablePrompt 0
[OK]
enablePrompt 1
[OK]
UT1>
```



7- **EnablePrompt <En>** can enable or disable prompt (enabled by default), see figure above

8- Readout quick start commands:

- UT1_EnTHGLG 1 1 1** to **configure SIM FEB data type sent** by enabling Timing, HG and LG on FEB simulator
- UT1_SetParam 64 4 258 1 16** to **configure SIM FEB** with an event having 64 channels enabled per GTS on 4 GTS (i.e. 256-ch a total sent over 4 GTS), Gate #=258 (fixed), 1 empty GTS at end of 4 GTS with data cycle and a span of 16 for timing and analog values
- UT1_SetDaqDelay 10** to **set a delay** of 10ms between each DAQ event sent over TCP
- UT1_EnReadout 1** to **start the readout** (DAQ client must be connected and listening to the data), OCB green led will be ON
- UT1_EnReadout 0** to **stop the readout** (DAQ client must be connected and listening to the data), OCB green led will be OFF

NB: span is made with a value added to the RT/FT/HG/LG with the GTS counter modulo span value set (see algorithm below). For instance:

- GTS/EVT=4, EmptyGTS=1 means 5 GTS sent per event, if SPAN=16 then the value added to RT/FT/HG/LG will be (#GTS+5) modulo 16 at each event sent
- GTS/EVT=1, EmptyGTS=0 means 1 GTS sent per, if SPAN=16 then the value added to RT/FT/HG/LG will be (#GTS+1) modulo 16 at each event sent

Appendix: Algorithm used by SIM FEB to simulate data sent

Modified version (red) for v.450 and higher

```
if(_enReadout)
{
    S_UInt32* l_ptr = _febBuffer[0][0];

    // latch the parameters for 1 event
    S_Byte l_chPerGts = _chPerGts;
    S_Byte l_gtsPerEvent = _gtsPerEvent;
    S_Byte l_gtsEmpty = _gtsEmpty;
    S_UInt16 l_gateNb = _gateNb;
    S_Byte l_spanGts = _spanGts-1;
    S_Boolean l_timeHgLg[3];
    l_timeHgLg[0] = _TimeHgLg[0];
    l_timeHgLg[1] = _TimeHgLg[1];
    l_timeHgLg[2] = _TimeHgLg[2];

    // OCB data packet header -----
    _ocbDataPacketHeader(&l_ptr, _GATE_TYPE, (S_Byte)l_gateNb, m_event); // Event header written by OCB PS
    _febGateHeader0(&l_ptr, _BOARD_ID, _GATE_TYPE, l_gateNb); // event always start with gate header written by OCB PL

    // FEB data per event starts here -----
    S_UInt16 l_wordSentQty = 0;

    _febGtsHeader(&l_ptr, &l_wordSentQty, m_gtsCnt);

    // #N GTS per Event
    for(S_Byte l_i=0; l_i<l_gtsPerEvent; l_i++)
    {
        // compute value added to T/HG/LG wrt to GTS = T/HG/LG span
        S_Byte l_span = m_gtsCnt & l_spanGts;

        // #N channels sent per GTS
        for(S_Byte l_j=0; l_j<l_chPerGts; l_j++)
        {
            S_Byte l_ch = ((l_i&0x3)<<6) | (l_j&0x3f);

            if(l_timeHgLg[_THGLG_TIME])
            {
                _febTime(&l_ptr, &l_wordSentQty, l_ch, 0, (S_Byte)(m_gtsCnt&0x3), _TIME_RISING, l_ch*3 + 16 + l_span);
                // l_span*2 to see span on Tdiff=TF-TR
                _febTime(&l_ptr, &l_wordSentQty, l_ch, 0, (S_Byte)(m_gtsCnt&0x3), _TIME_FALLING, (S_UInt16)l_ch*2 + 2000 + l_span*2);
            }

            if(l_timeHgLg[_THGLG_HG])
                _febAmplitude(&l_ptr, &l_wordSentQty, l_ch, 0, (S_Byte)(m_gtsCnt&0x3), _AMPL_ID_HG, 0xFFF-l_ch*3 + l_span);

            if(l_timeHgLg[_THGLG_LG])
                _febAmplitude(&l_ptr, &l_wordSentQty, l_ch, 0, (S_Byte)(m_gtsCnt&0x3), _AMPL_ID_LG, 0xFF-l_ch + l_span);
        }

        _febGtsTrailers(&l_ptr, &l_wordSentQty, m_gtsCnt, true);
        m_gtsCnt++;
        _febGtsHeader(&l_ptr, &l_wordSentQty, m_gtsCnt);
    }

    // add empty GTS
    for(S_Byte l_i=0; l_i<l_gtsEmpty; l_i++)
    {
        _febGtsTrailers(&l_ptr, &l_wordSentQty, m_gtsCnt, false);
        m_gtsCnt++;
        _febGtsHeader(&l_ptr, &l_wordSentQty, m_gtsCnt);
    }

    _febEventDone(&l_ptr, _BOARD_ID, (S_Byte)l_gateNb, l_wordSentQty); // last word sent by FEB
    _febGtsTrailers(&l_ptr, &l_wordSentQty, m_gtsCnt, false);
    m_gtsCnt++;
    _febDataPacketTrailer(&l_ptr, _BOARD_ID, 0, 0); // Word added by OCB for FEB : no OCB errors during FEB data packet
    // FEB data per event ends here -----

    _ocbDataPacketTrailer(&l_ptr, _GATE_TYPE, (S_Byte)l_gateNb, 0, 0); // Event trailer built by OCB PS : no OCB errors during event
    // OCB data packet ends -----

    S_Int l_nwrote = m_com.halWrite((char*)_febBuffer[0][0], (l_wordSentQty+_OCB_PACKET_W32_NB)*4);
    if (l_nwrote < 0)
        return false;

    m_event++;
}

if(_resetGts)
{
    m_gtsCnt = 0;
    m_event = 0;
    _resetGts = false;
}

S_taskDelay(_daqDelay);
```

For an event with 2 GTS with data & 1 GTS without any data, the correct sequence is now:

Event #0:

OCBDataHeader
GateHeaderA
GTSHeader
Data ...
GTSTrailer1
GTSTrailer2 (Data)
GTSHeader
Data ...
GTSTrailer1
GTSTrailer2 (Data)
GTSHeader
GTSTrailer1
GTSTrailer2 (No Data)
GTSHeader
EventDone
GTSTrailer1
GTSTrailer2 (No Data)
OCBFebDataTrailer
OCBDataTrailer

Event #1:

OCBDataHeader
GateHeaderA
GTSHeader
Data ...
...

EventDone mismatch explanation:

- EventDone is sent by the FEB with the #words sent from GATE or EVENT message.
- In continuous GATE opening (i.e. like in FASER CAL), the gate message sent by the FEB is only sent once, basically at the start of the RUN, so if we have a coincidence right after the gate opening (unlikely), we got the correct #words in EventDone because the FEB will send 3 consecutive words (Gate header 1 + Gate header 2 + Gate Time) : see FEB protocol
- However, if an EVENT message is received at least 1 GTS after the gate opening, the OCB artificially produce a GATE header 0 / FEB #x in order to notify a FEB data packet start for the FEB #x, Gate header 2 + Gate Time will never be sent in this case

Check to do on Gate word(s):

- only 1 word (GATE header 0 = artificial OCB):
EventDone #Word = FEB data packet length - 1
- 3 words (Gate header 1 + Gate header 2 + Gate Time):
EventDone #Word = FEB data packet length