Aurix™ Embedded Automotive COM protocols

Nicolò Spagnolo (DC ATV MC TM CES3)
27/03/2024

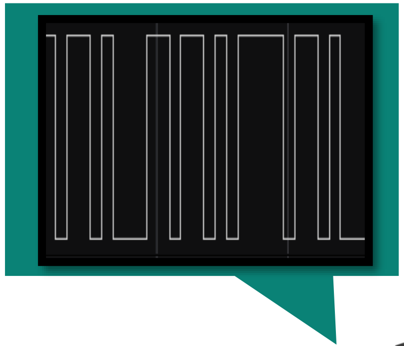
Introduction

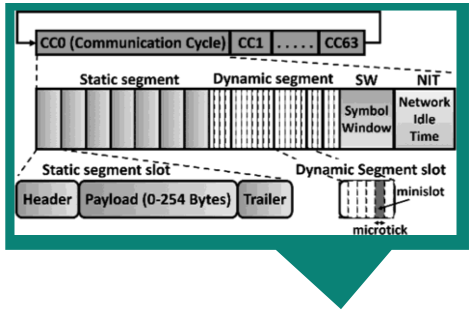
Type of COM protocols in Automotive

- UART
- SPI
- LIN
- CAN/CANFD
- ETH
- Flexray
- I2C
- I2S
- SENT
- PSI5

SAE INTERNATIONAL®

01010111 01101000
01100001 01110100
00100111 01110011
00100000 01110101
01110000 00111111
00001010



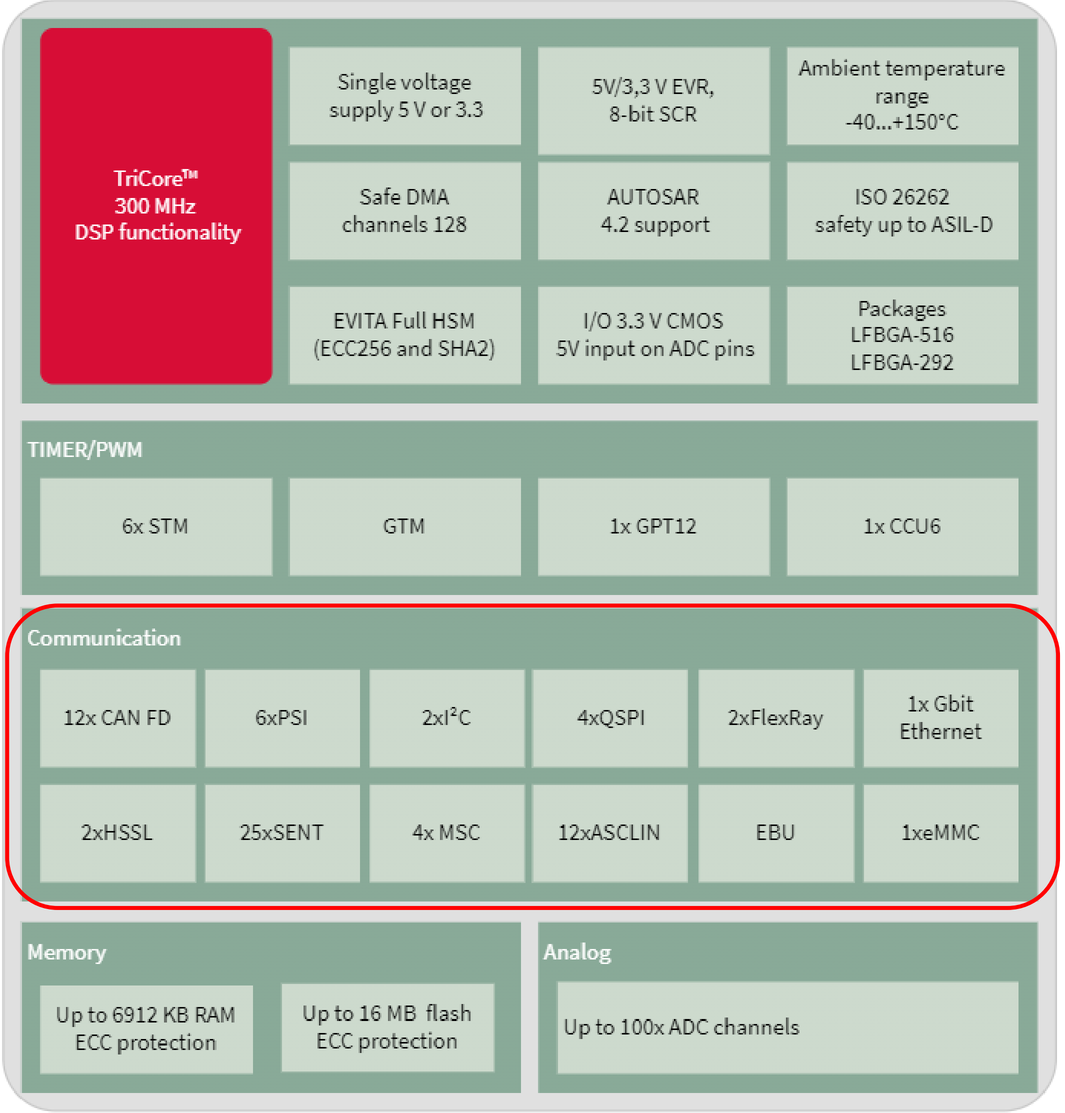


Comparison between Automotive COM protocols

Parameters	UART	LIN	CAN	FlexRay
Architecture	Two devices	Single master and up to 15 slaves	Multiple nodes (20, 32)	Multiple nodes (up to 64)
Topology	Direct connection	Bus topology	Bus topology	Bus/Star topology
Message transmission	Asynchronous	Synchronous	Asynchronous	Synchronous/Async
Data rate or Baud rate	Max typ ≈ 115kbps	Max. 20kbps	Max. 1Mbps	Max. 10Mbps
Error checking mechanism	Parity bit	Checksum over the Protected Identifier and Data fields	CRC computation over the entire frame	Two CRC computations
Physical layer	Single electrical wire	Single electrical wire	Electrical dual wire	Dual wire
Cabling impedance	/	1k ohms	120 ohms	80-110 ohms
Range	/	1-5 kilometers	40 meters	10 meters

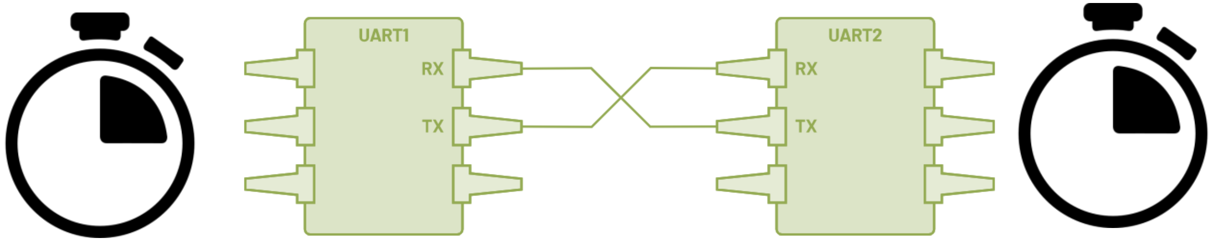
Type of COM protocols in AURIX™

– Aurix™ TC39x Architecture



UART Communication – Clocking system

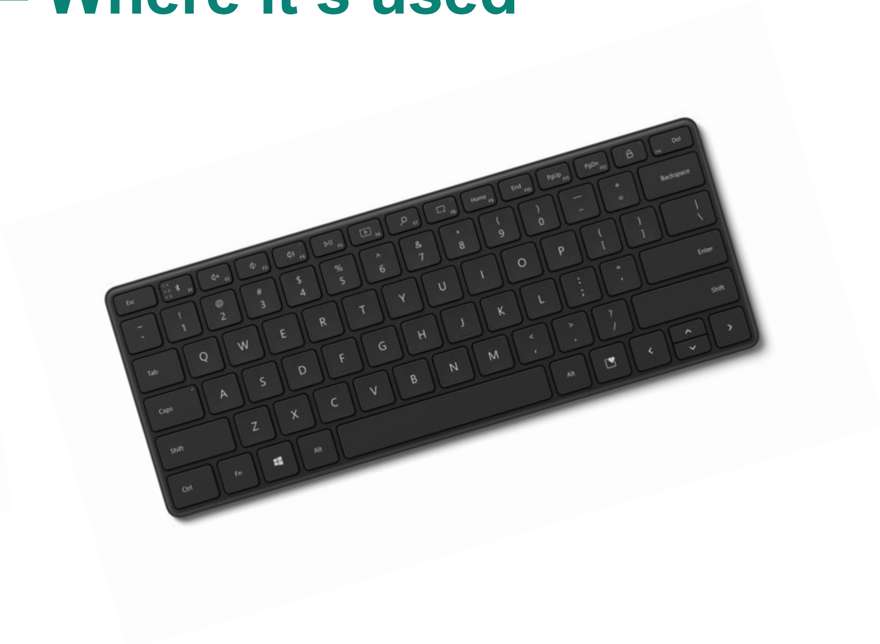
- UART stands for Universal Asynchronous Receiver-Transmitter and refers to an electronic module capable of communicating **asynchronously** with another module by both transmitting and receiving data
- By "asynchronous", as we shall see, we mean the characteristic of not having synchronism between the two modules, there is no particular signal that keeps the modules synchronized with each other. This has both positive (simplicity) and negative (possibility of frequency drift) implications. To guarantee correct communication **the baudrate of the two devices must be set to the same value**

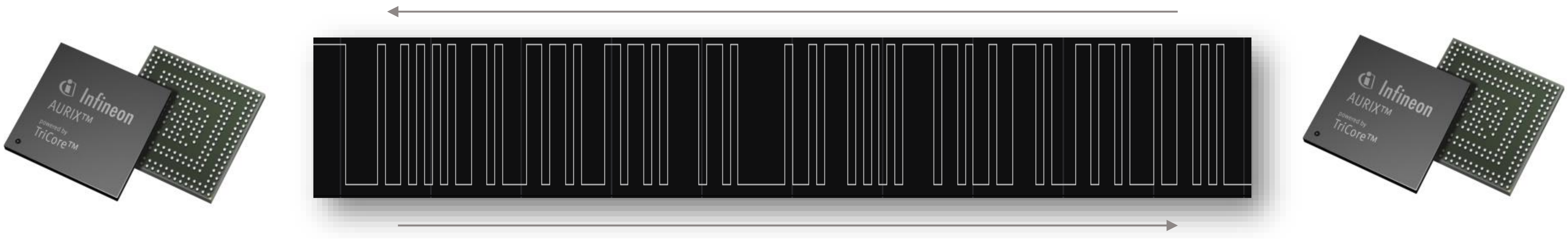


Baudrate = 9600

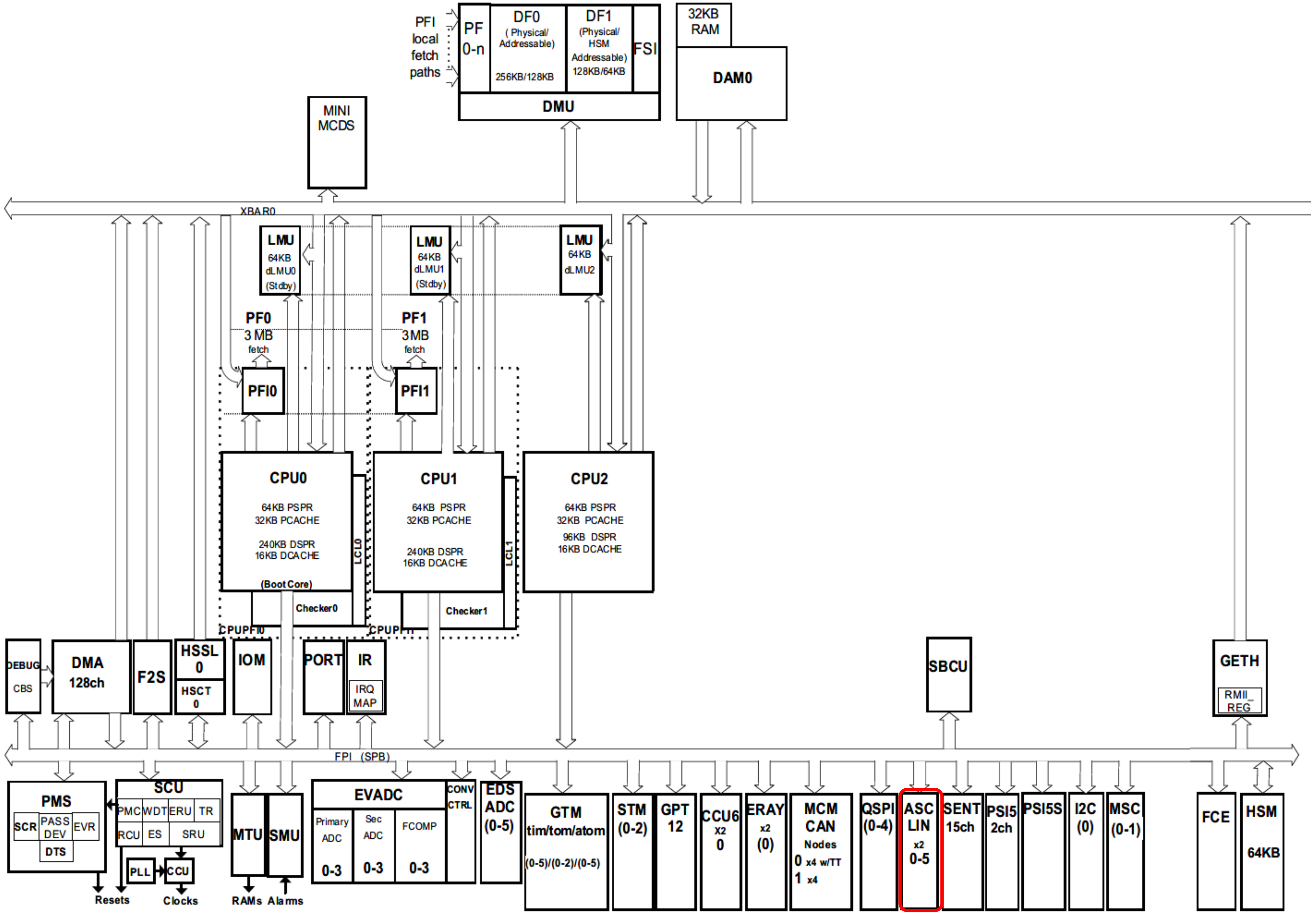
Baudrate = 9600

UART Communication – Where it’s used





AURIX™ TC37x - Block Diagram



restricted

AURIX™ TC37x – ASCLIN module

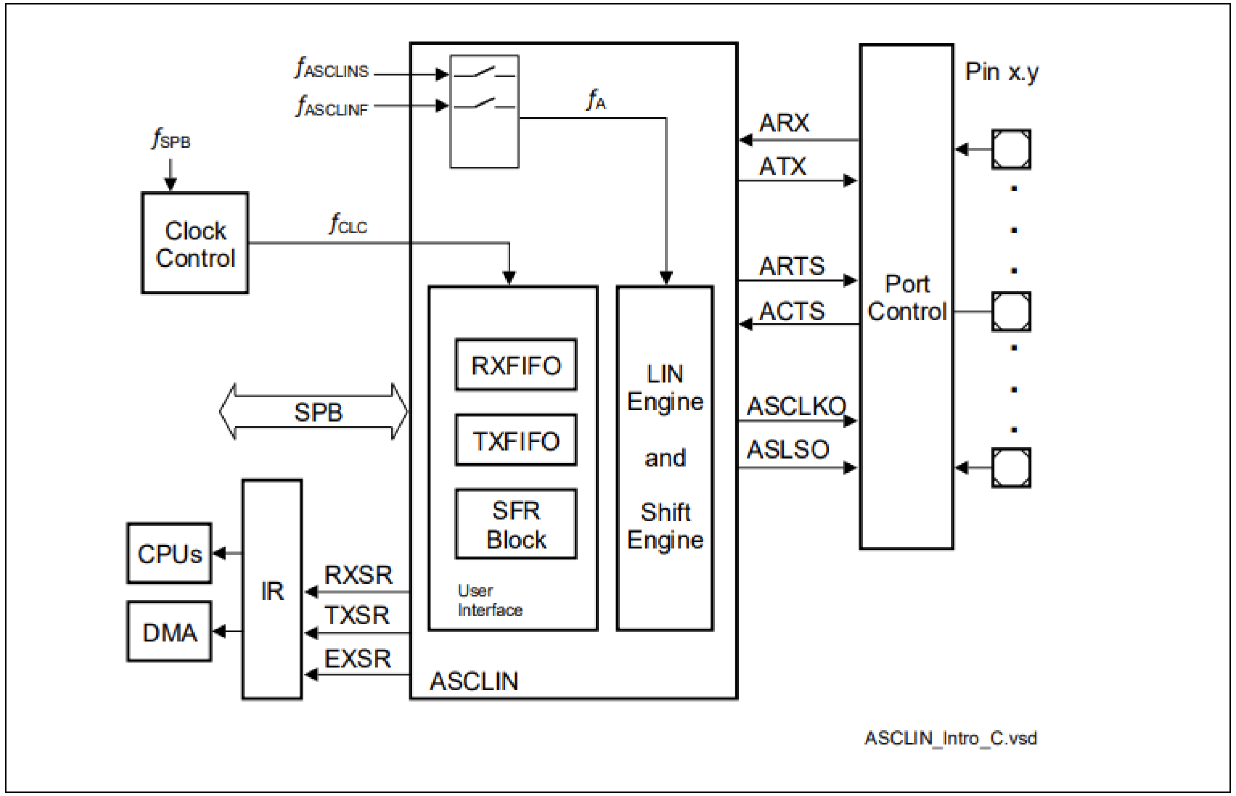
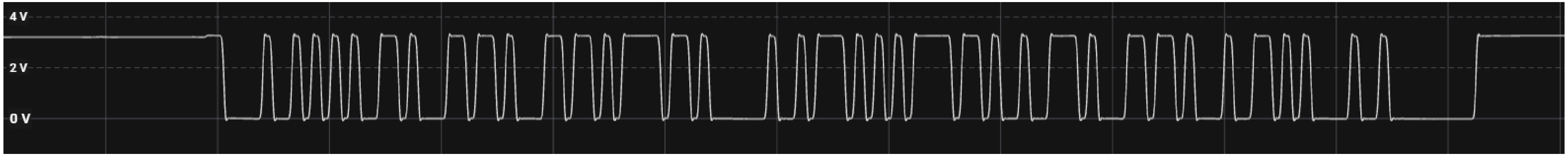


Figure 415 Block Diagram of the ASCLIN module.

Standard ASC Features

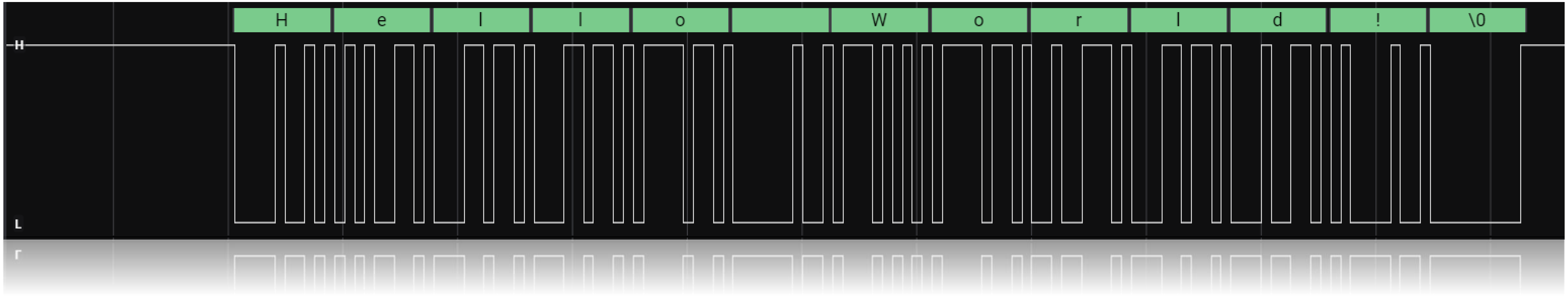
- Full-duplex asynchronous operating mode
  - 7-bit, 8-bit or 9-bit (or up to 16-bit) data frames, LSB first
  - Parity-bit generation/checking
  - One or two stop bits
  - Max baud rate f_A / 16 (6.25 MBaud @ 100 MHz f_A module clock)
  - Min. baud rate f_A / 268 435 456 (0.37 Baud @ 100 MHz f_A module clock)
- Optional RTS / CTS handshaking

How UART frame is composed?



Analog Signal

Digital Signal



Physical layer

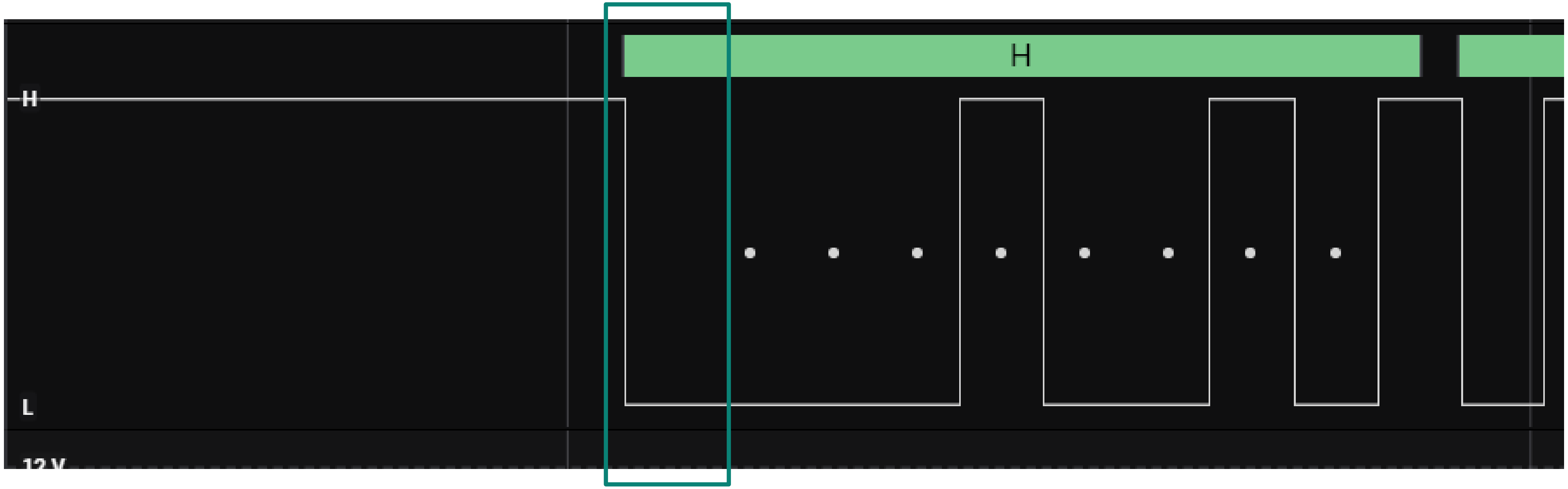
– How the analog signal can be interpreted by the microcontroller? According to DS

Input high voltage level	V_{IH} SR	0.7 * $V_{EXT/FLEX/EVRSB}$	-	-	V	AL
		2.0	-	-	V	TTL
Input low voltage level	V_{IL} SR	-	-	0.44 * $V_{EXT/FLEX/EVRSB}$	V	AL

– µC recognize a digital «1» when the voltage level exceeds $0{,}7 * V_{EXT} = 3{,}5V$ (minimum)
– µC recognize a digital «0» when the voltage level exceeds $0{,}44 * V_{EXT} = 2{,}2V$ (maximum)

*VEXT = 5V in our case

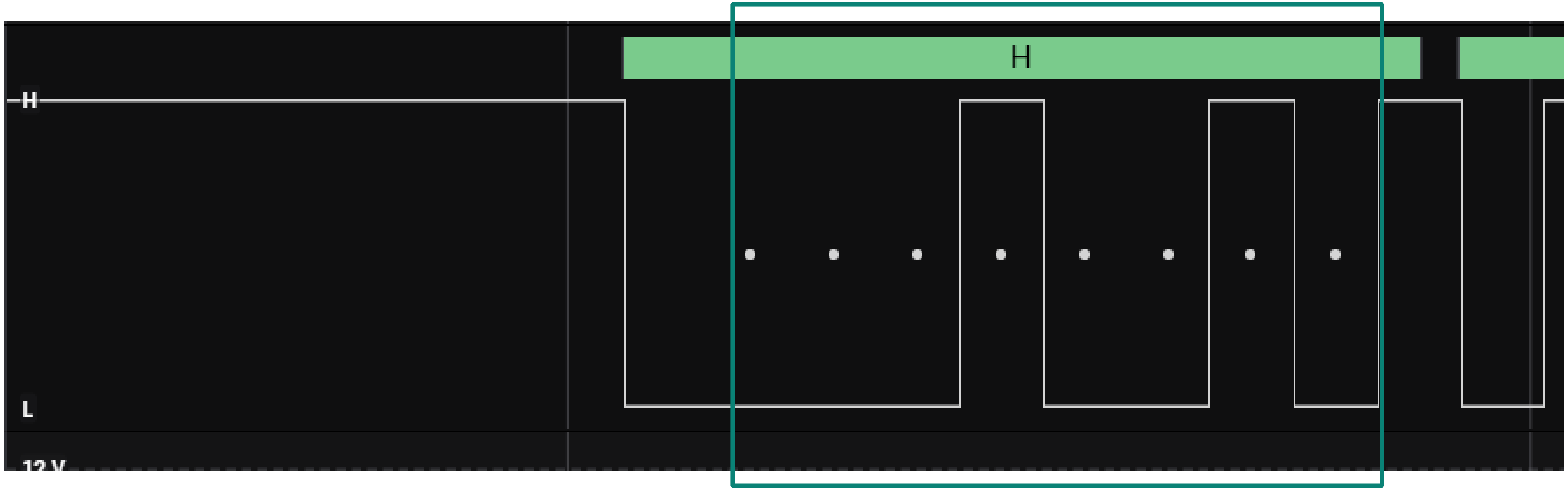
Start bit



$$t\_bit = \frac{1}{Baudrate} = \frac{1}{115200} \approx 8{,}7\mu s$$

› Start bit: At the beginning of a UART frame, signal stays in idle mode (high logic). When the device send the message the first bit is always a «0»

Data field

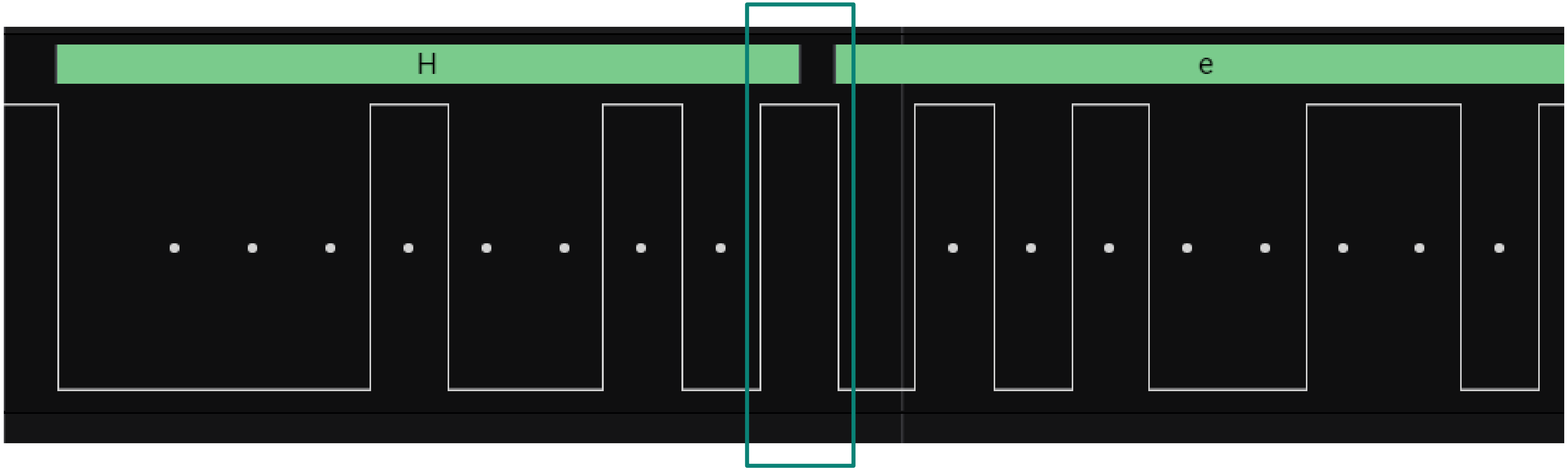


› Data: In this frame, the first data transmitted is the letter "H"

0100 1000 → H

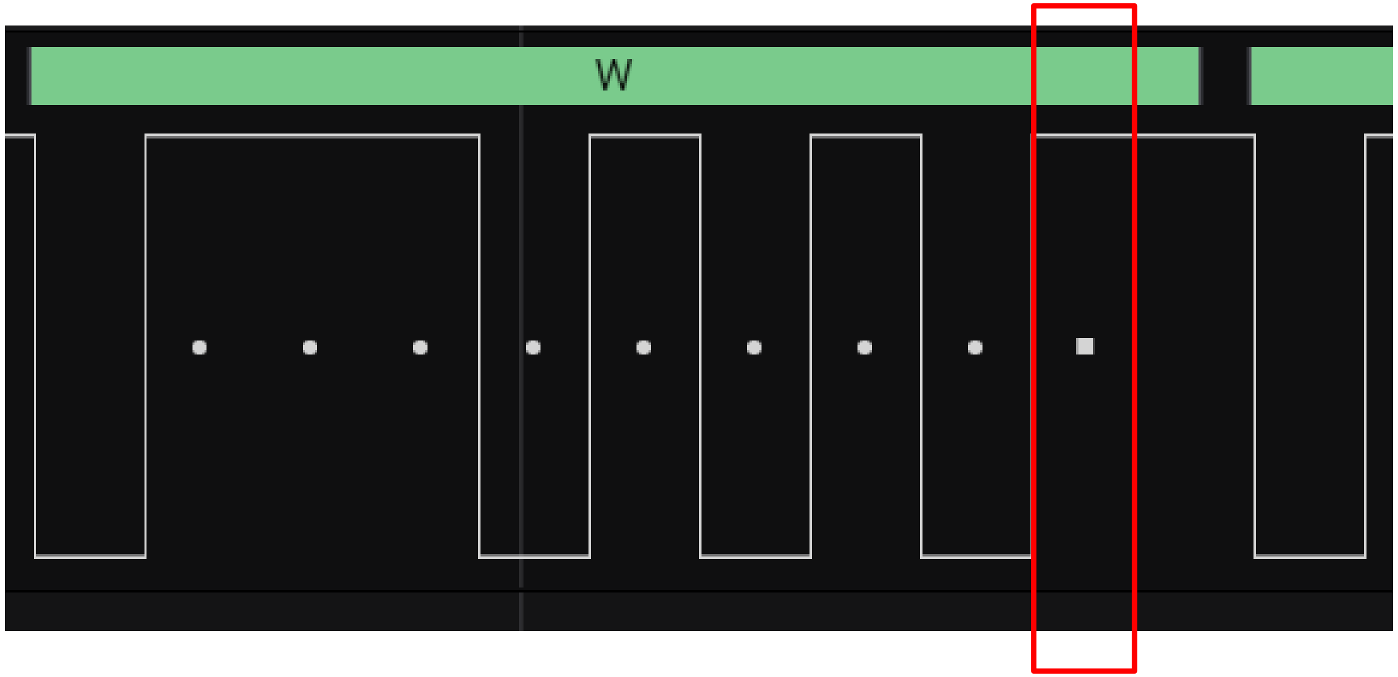
Binary → ASCII

Stop bit



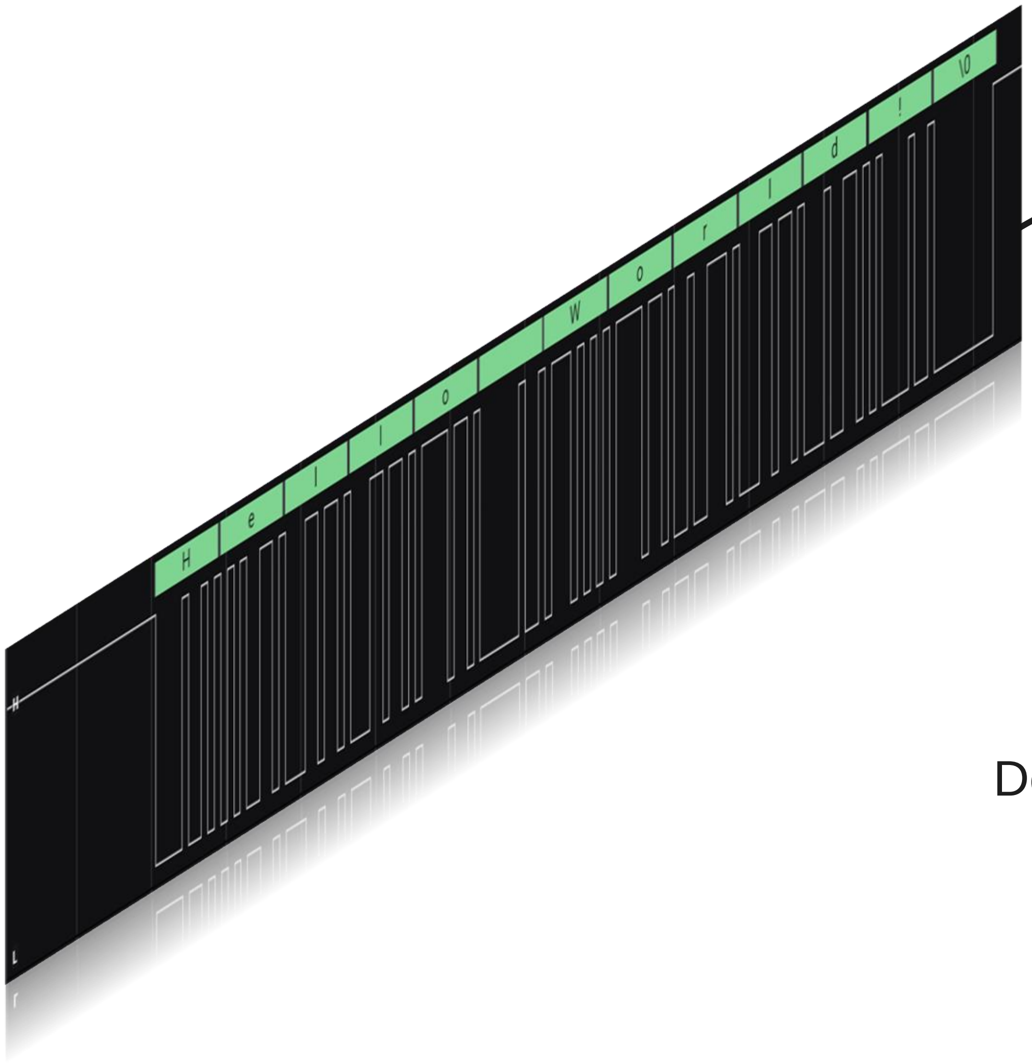
› Stop Bit: After the 8 bits of data, UART frame ends with one or two stop bits. In this example we have only one stop bit followed by the start bit of the following data frame

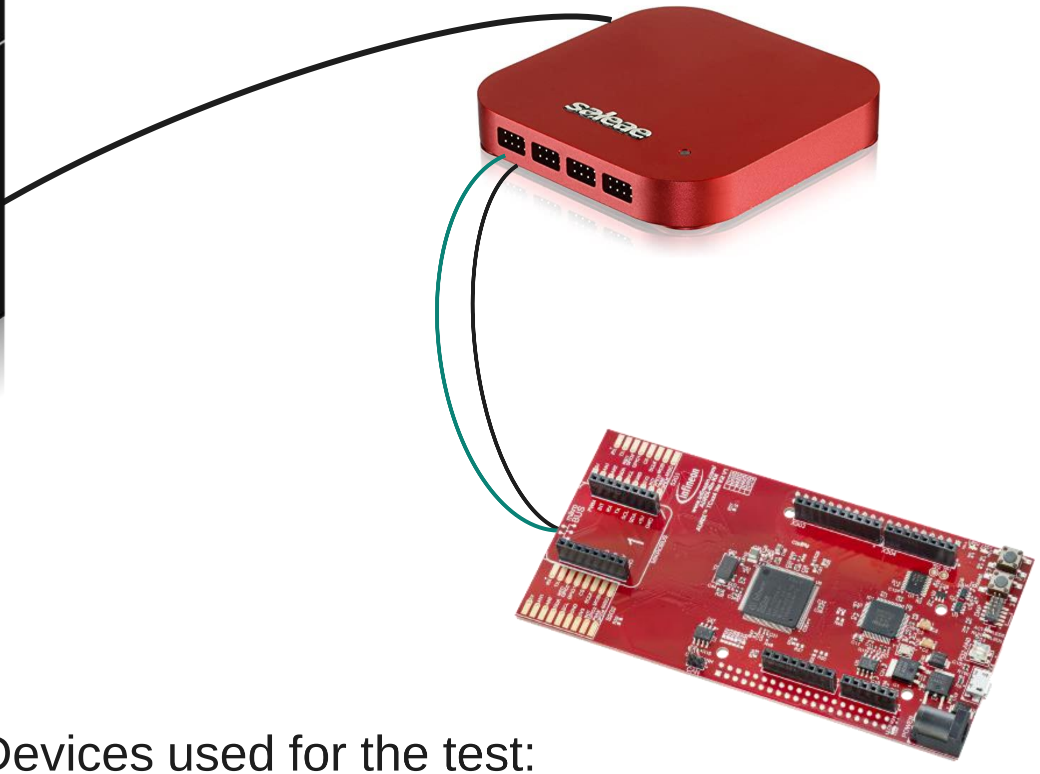
Frame with one Parity Bit



› Checksum: In this case the letter "W" is encoded with 0101 0111 (LSB) and counting the HIGH states (the "1") we realize that they are in an **odd** number. Here comes the parity bit that before the stop bit brings the state to HIGH so that the number of HIGHs in the frame is finally **even**

UART protocol seen by oscilloscope – Test with AURIX™

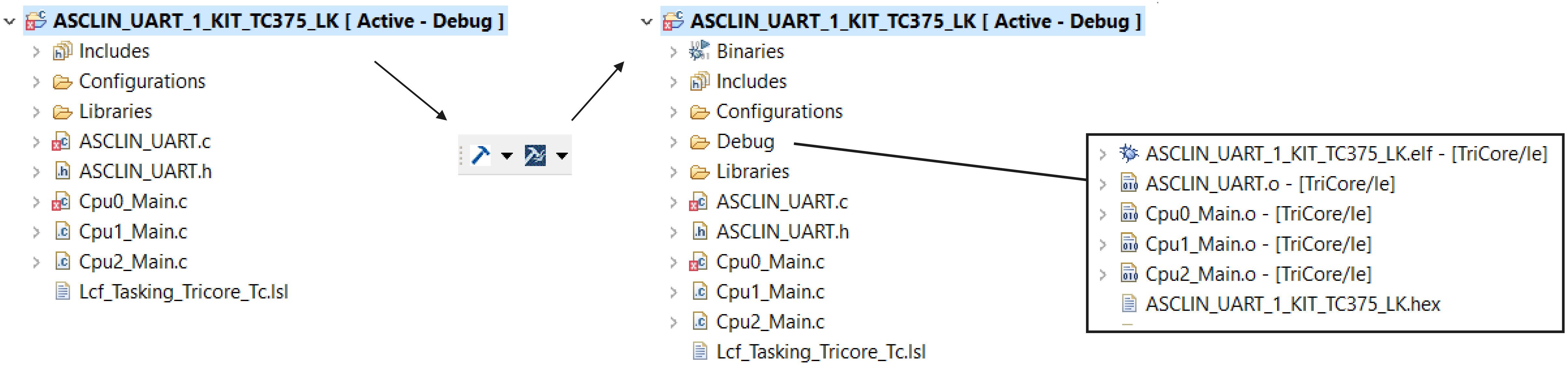




Devices used for the test:

- Aurix TC375LK
- Saleae Logic Pro 16 (Digital oscilloscope)
- AURIX™ Development Studio

ASCLIN_UART example



Let's get into the code

Cpu0_Main.c

```c
 * \lastUpdated 2021-03-22
 *********************************************************************************
#include "Ifx_Types.h"
#include "IfxCpu.h"
#include "IfxScuWdt.h"
#include "ASCLIN_UART.h"

IFX_ALIGN(4) IfxCpu_syncEvent g_cpuSyncEvent = 0;

void core0_main(void)
{
    IfxCpu_enableInterrupts();

    /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
     * Enable the watchdogs and service them periodically if it is required
     */
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
    IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());

    /* Wait for CPU sync event */
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 1);

    init_ASCLIN_UART();               /* Initialize the module                 */
    IfxCpu_enableInterrupts();        /* Enable interrupts after initialization */
    send_receive_ASCLIN_UART_message(); /* Send the string                     */

    while(1)
    {
    }
}
```

Cpu1_Main.c

```c
 * machine-executable object code generated by a source language processor.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR I
 * WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND N
 * COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMA
 * CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * IN THE SOFTWARE.
 *********************************************************************************
#include "Ifx_Types.h"
#include "IfxCpu.h"
#include "IfxScuWdt.h"

extern IfxCpu_syncEvent g_cpuSyncEvent;

void core1_main(void)
{
    IfxCpu_enableInterrupts();

    /* !!WATCHDOG1 IS DISABLED HERE!!
     * Enable the watchdog and service it periodically if it is required
     */
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());

    /* Wait for CPU sync event */
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 1);

    while(1)
    {
    }
}
```

Pinout

– What does it mean?

```
#define UART_PIN_RX             IfxAsclin1_RXB_P15_5_IN             /* UART receive port pin          */
#define UART_PIN_TX             IfxAsclin1_TX_P15_5_OUT             /* UART transmit port pin         */
```

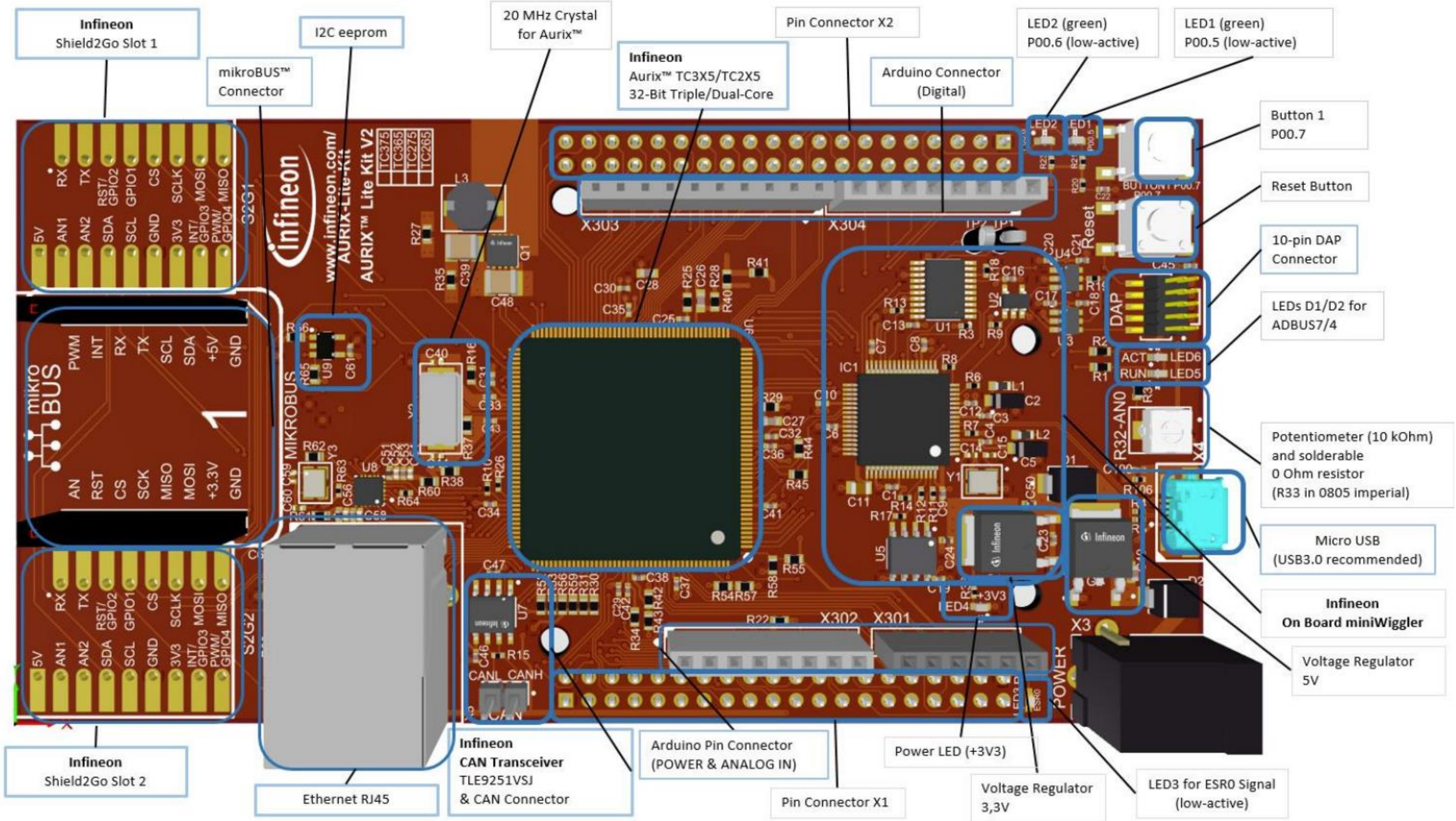
Pinout

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	NC1	VEXT	P10.7	P10.6	P10.2	P10.3	P10.0	P11.11	P11.9	P11.2	P13.3	P13.1	P14.8	P14.5	P14.1	P15.6	P15.4	P15.1	VDDP3	VSS
B	P02.0	VSS	VEXT	P10.8	P10.5	P10.4	P10.1	P11.12	P11.10	P11.3	P13.2	P13.0	P14.6	P14.3	P14.4	P14.0	P15.3	VDDP3	VSS	P15.0
C	P02.2	P02.1																	P15.2	P20.14
D	P02.4	P02.3		VSS	VFLEX	P11.15	P11.14	P11.5	P11.6	P11.4	P14.10	P14.9	P14.7	P15.8	P15.7	VDD	VSS		P20.12	P20.13
E	P02.6	P02.5		P02.9	VSS	P11.13	P11.8	P11.7	P11.1	P11.0	P12.1	P12.0	P14.2	P15.5	VDD	VSS	P20.9		P20.10	P20.11
F	P02.8	P02.7		P02.11	P02.10											ESR0	P20.6		P20.7	P20.8
G	P00.0	P00.1		P01.4	P01.3			VDDSB (VDD)	VSS	DAPE2	DAPE1	VSS	VDD			ESR1	PORST		P20.1	P20.3
H	P00.2	P00.3		P01.6	P01.5		VDDSB (VDD)		VSS	VSS	VSS	VSS		VDD		P21.7 / TDO	P21.6 / TDI		P20.2	P20.0
J	P00.4	P00.5		P00.6	P01.7		VSS	VSS		VSS	VSS		VSS	VSS		TCK	P21.1		P21.3	P21.5
K	P00.7	P00.9		P00.8	P00.10		AGBTC LKP (VSS)	VSS	VSS	VSS	VSS	VSS	VSS	DAPE0		TMS	P21.0		P21.2	P21.4
L	P00.11	P00.12		AN43	AN42		AGBTC LKN (VSS)	VSS	VSS	VSS	VSS	VSS	VSS	AGBTE RR (VSS)		P22.10	P22.11		TRST	VSS
M	AN46	AN47		AN41	AN40		VSS	VSS		VSS	VSS		VSS	VSS		P22.8	P22.9		XTAL2	XTAL1
N	AN44	AN45		AN36 / P40.6	AN38 / P40.8		VDD		VSS	VSS	VSS	VSS		VDD		P22.6	P22.7		VDD	VEXT
P	AN39 / P40.9	AN37 / P40.7		AN32 / P40.4	AN34			VDD	VSS	AGBTT XN (VSS)	AGBTT XP (VSS)	VSS	VDD			P22.4	P22.5		P22.1	P22.0
R	AN33 / P40.5	AN35		AN31	AN23											P23.7	P23.6		P22.3	P22.2
T	VAREF 2	VAGND 2		AN30	AN22	AN15	AN12	AN6	AN4	AN0	VEVRS B	P34.2	P34.4	P33.14	P32.5	VSS	P23.5		P23.3	P23.4
U	AN29 / P40.14	AN28 / P40.13		NC1	AN17 / P40.10	AN14	AN9	AN7	AN3	AN1	P34.1	P34.3	P34.5	P33.15	P32.6	P32.7	VSS		P23.1	P23.2
V	AN27 / P40.3	AN26 / P40.2																	VEXT	P23.0
W	AN25 / P40.1	AN24 / P40.0	AN19 / P40.12	AN18 / P40.11	AN16	AN13	AN11	AN8	AN2	P33.0	P33.2	P33.4	P33.6	P33.8	P33.10	P33.12	P32.1 / VGATE 1P	P32.4	VSS	VEXT
Y	NC1	AN21	AN20	VSSM	VDDM	VAREF 1	VAGND 1	AN10	AN5	P33.1	P33.3	P33.5	P33.7	P33.9	P33.11	P33.13	P32.0 / VGATE 1N	P32.2	P32.3	VSS

TC39xed - (top view)

Ball	Symbol	Ctrl.	Buffer Type	Function
E14	P15.5	I	FAST / PU1 / VEXT / ES	**General-purpose input**
	GTM_TIM3_IN0_4			**Mux input channel 0 of TIM module 3**
	GTM_TIM2_IN0_4			**Mux input channel 0 of TIM module 2**
	ASCLIN1_ARXB			**Receive input**
	I2C0_SDAC			**Serial Data Input 2**
	QSPI2_MTSRA			**Slave SPI data input**
	SCU_E_REQ4_3			**ERU Channel 4 inputs 0 to 5 (0 is the LSB and 5 is the MSB)**
	P15.5	O0		**General-purpose output**
	GTM_TOUT76	O1		**GTM muxed output**
	ASCLIN1_ATX	O2		**Transmit output**
	IOM_MON2_13			**Monitor input 2**
	IOM_REF2_13			**Reference input 2**
	QSPI2_MTSR	O3		**Master SPI data output**
	—	O4		**Reserved**
	MSC0_EN0	O5		**Chip Select**
	I2C0_SDA	O6		**Serial Data Output**
	CCU60_CC61	O7		**T12 PWM channel 61**
	IOM_MON1_1			**Monitor input 1**
	IOM_REF1_5			**Reference input 1**

TC375LiteKit board



TC375LK

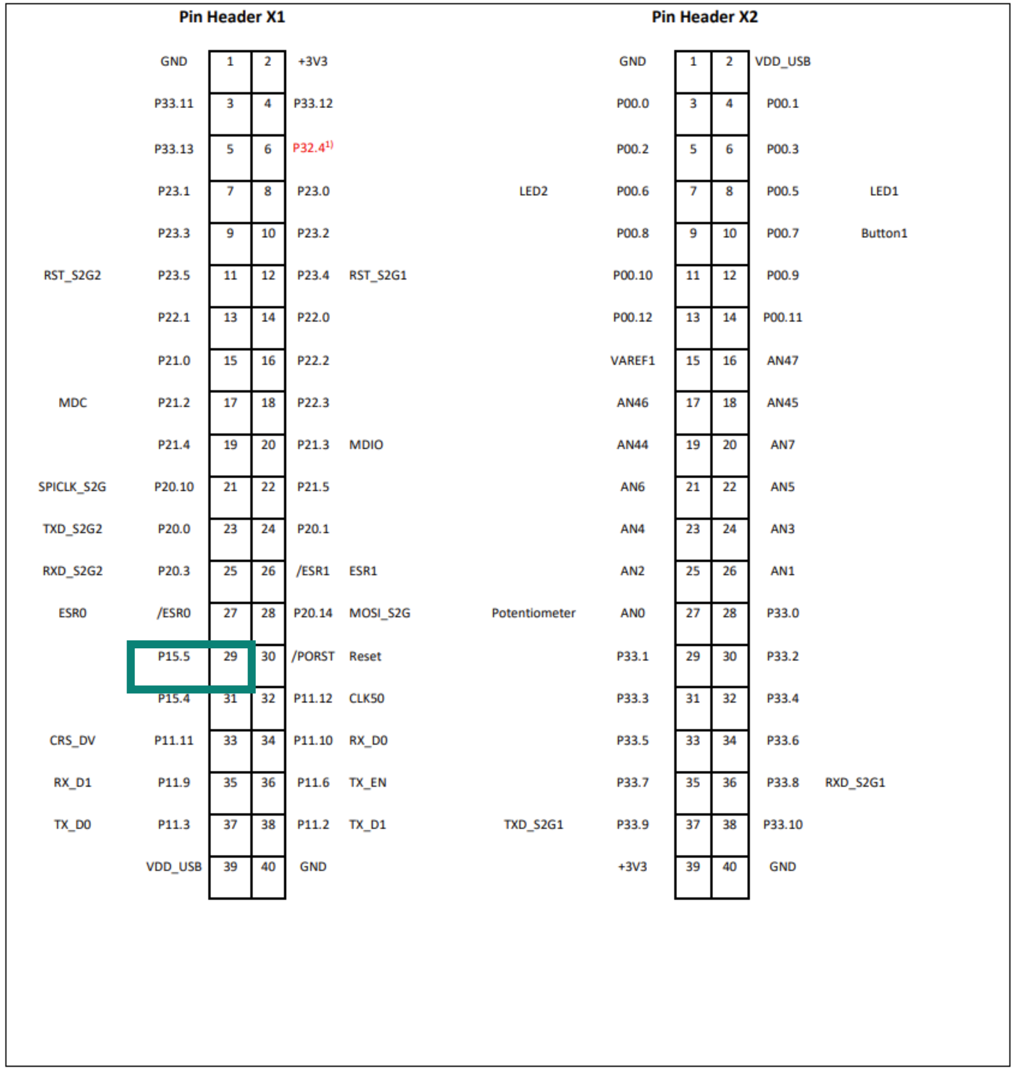
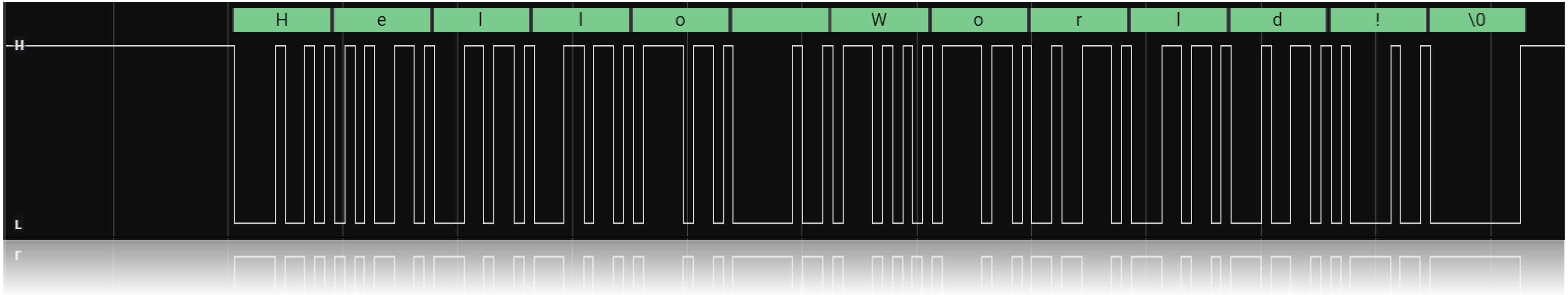


Figure 5 Signal mapping of the pin headers X1 and X2

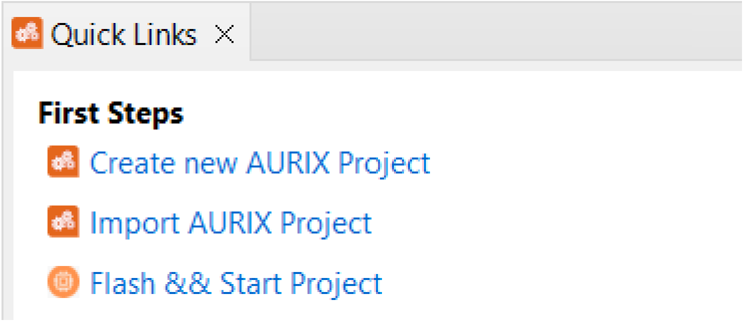
UART protocol seen by oscilloscope – Test with AURIX

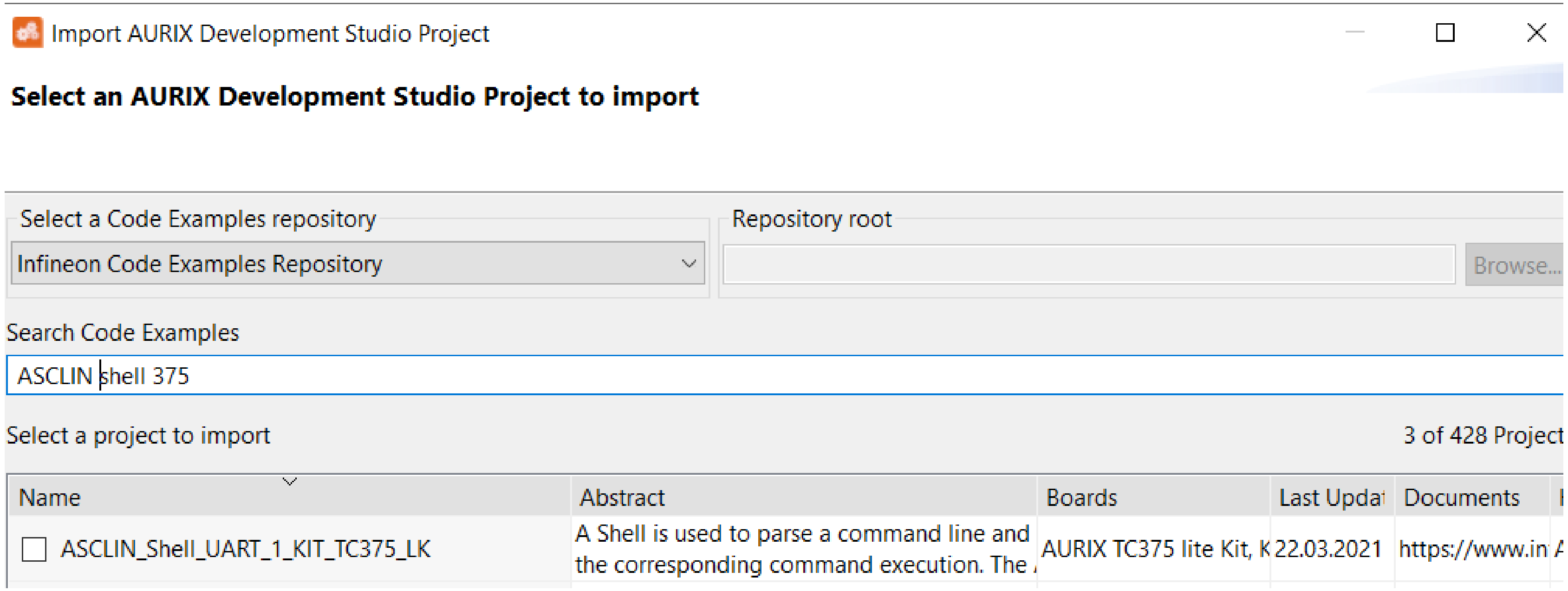


– The “ASCLIN_UART_1 for KIT_AURIX_TC375_LK” script has been loaded on TC375LK. Through the digital analyser it is possible to decode the output signal from the board. Specifically, this script allowed TC375LK to be used as a master in UART communication by sending the “Hello world!” message. (the figure shows the initial part of the message)

That’s your turn!

- Open Aurix Development Studio
- Press «Import AURIX Project»
- Search for «ASCLIN_Shell_UART_1_KIT_TC375_LK» and select it
- Press finish





First task

- The code example is ready to toggle (change the status) the ports where two LEDs are connected
- The UART communication is provided using ASCLIN0 module and the physical connection is routed through the USB port
- The terminal shows the messages sent by the microcontroller to the PC and you can send back too some commands
- The goal is to modify the code in order to have the possibility to **send a command to Aurix forcing the status of both LEDs to turn them OFF**

Second task

- The code example is ready to toggle (change the status) the ports where two LEDs are connected
- The UART communication is provided using ASCLIN0 module and the physical connection is routed through the USB port
- The terminal shows the messages sent by the microcontroller to the PC and you can send back too some commands
- The goal is to modify the code in order to have the possibility to **turn on both LEDs with a button available in the board**

infineon