



# Intel-FPGA as a Companion Chip

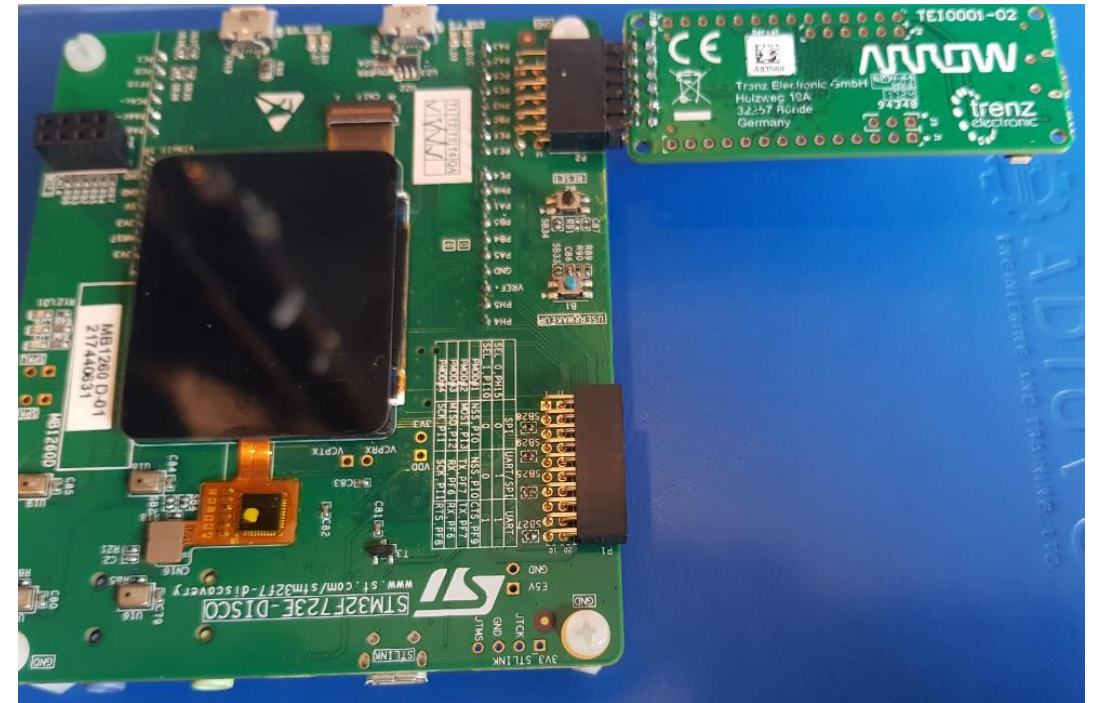
Adam Taylor

# Objective

To enable a traditional microprocessor to be able to access non-standard interfaces.

Use a low-cost Intel-FPGA board to translate from a processor supported interface e.g. SPI, I2C, QSPI, OSPI to one which is not for example analogue, GPIO, Motor Control, Cameras, Displays etc.

For this example, we are using Max1000 board and STM32F723E Disco board



# Why is a companion need

Several reasons a companion chip may be required, driven by system and interfacing requirements.

- Interface Translation – Customized / Bespoke / legacy interface
- Interface Expansion – Providing additional standard interfaces
- Deterministic response – Motor control, Servo Positioning
- Offload signal processing – Filtering / noise removal of ADC signals
- Leverage parallelism in the Intel FPGA – Process multiple signals / sensors simultaneously.

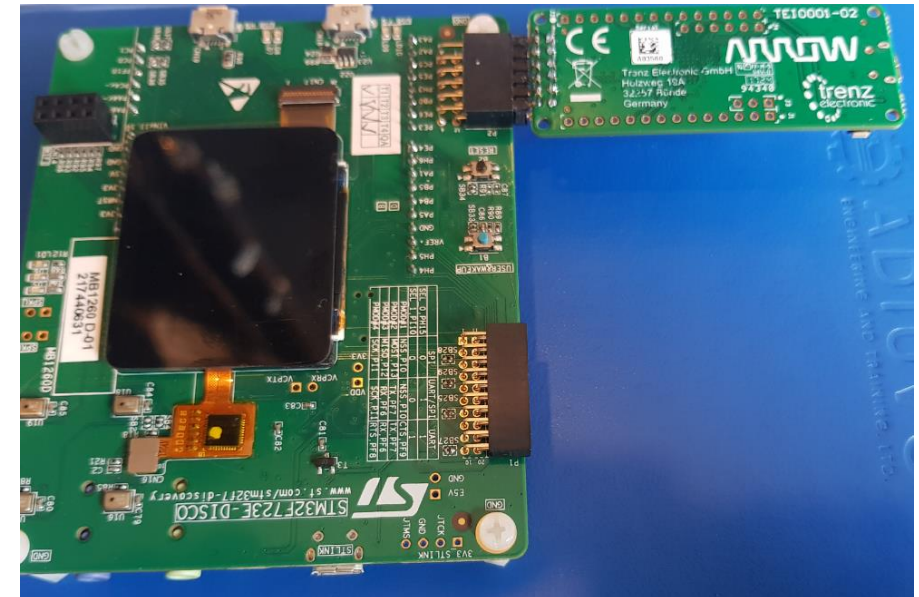
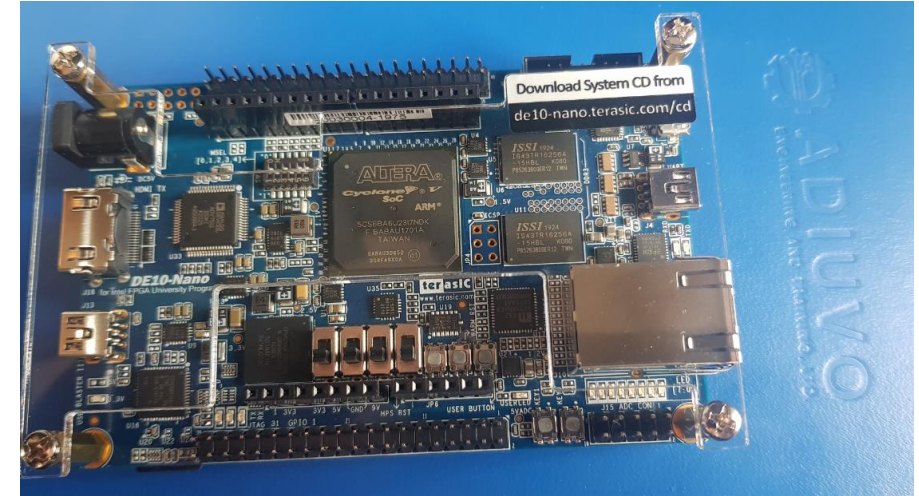
# Why not use a SoC

SoC combine Arm processors (A9, A53 etc.) with programmable logic.

Many developments may use a non-Arm processor and still need expansion for example Intel Atom

Logic resources might be very low depending on interface translation, making SoC overkill.

Ultimately at the end of the day the decision SoC or Intel FPGA and Companion board depends upon the system requirements.

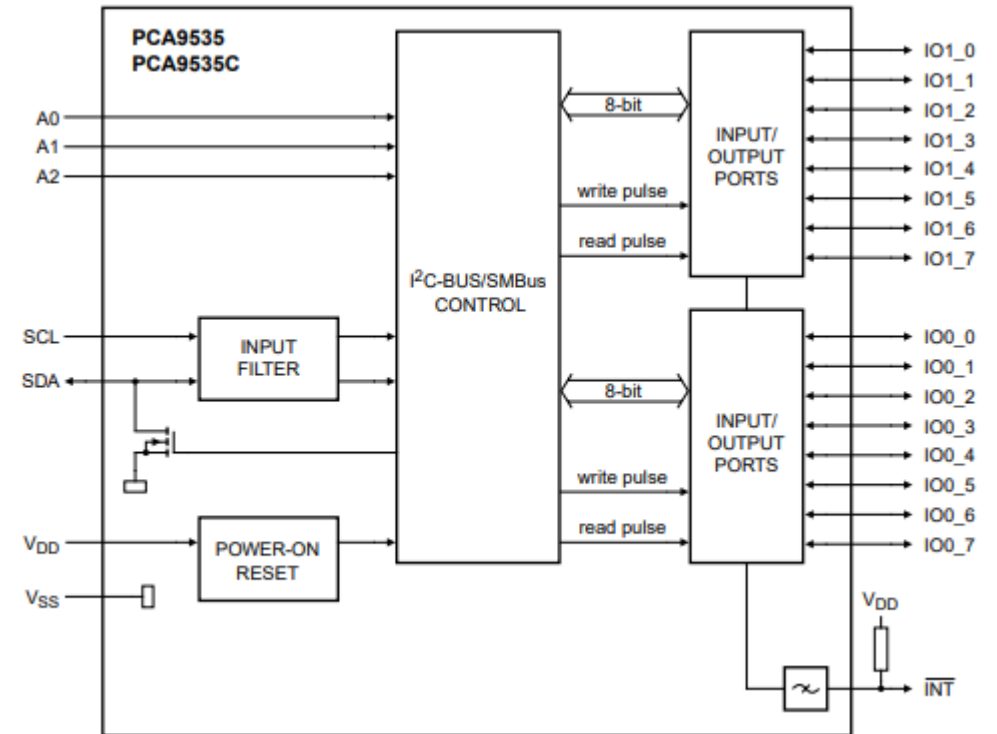


# Why not use a simple IO Expander

Simple expanders for example I2C have no intelligence  
Presented solution enables direct access to Avalon  
Master (via SPI/I2C)

Enables not only interfacing solutions but also

- Ability to monitor system health using inbuilt ADC
- Embedded signal processing – Filtering simple rolling average to FIR
  - Reduces processing load on the software – Software can work with the answer
  - Provides a more deterministic response
- Co-Processor could be used if desired for example



# The Max1000 Board

Board is small footprint with flexible IO

Low-cost device

Low Overhead to include in design  
either board or Max10 Device

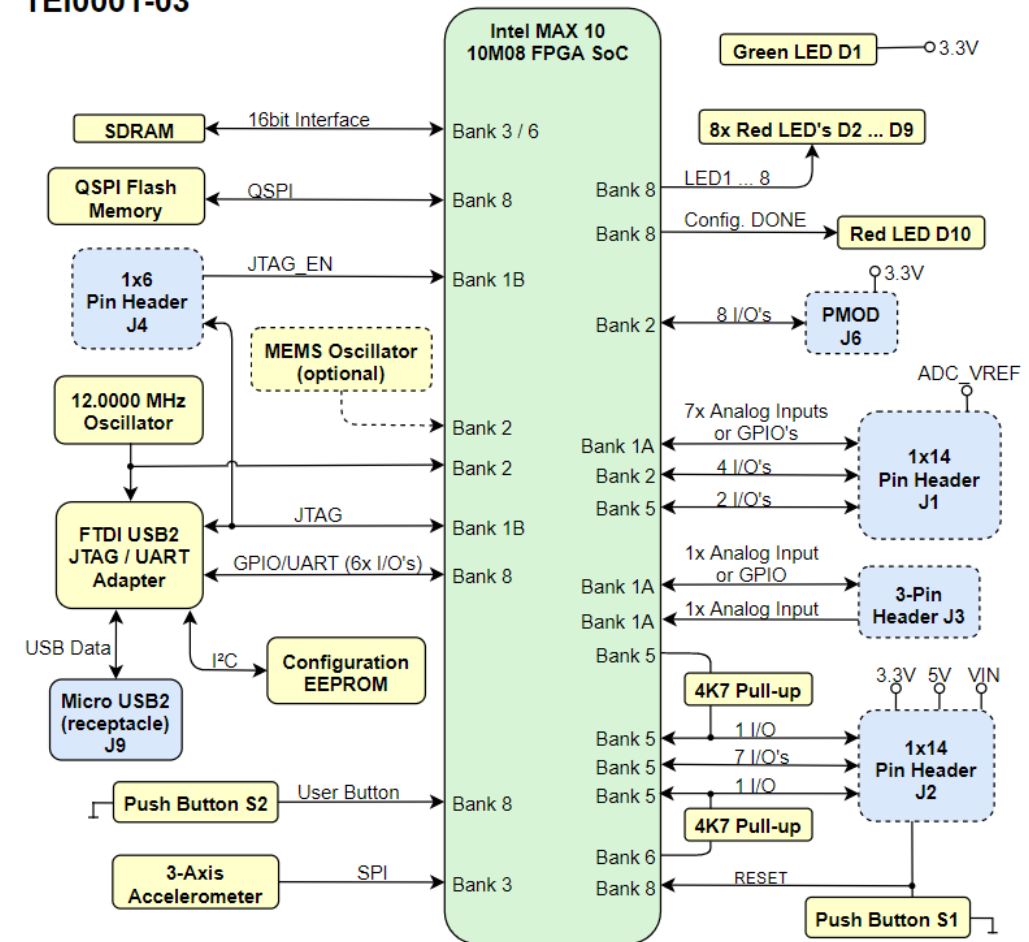
Flexible IO – provides a range of  
options

Breaks out most IO

- Provides simple LED / SW
- ADC

Ideal for demonstration applications

TEI0001-03

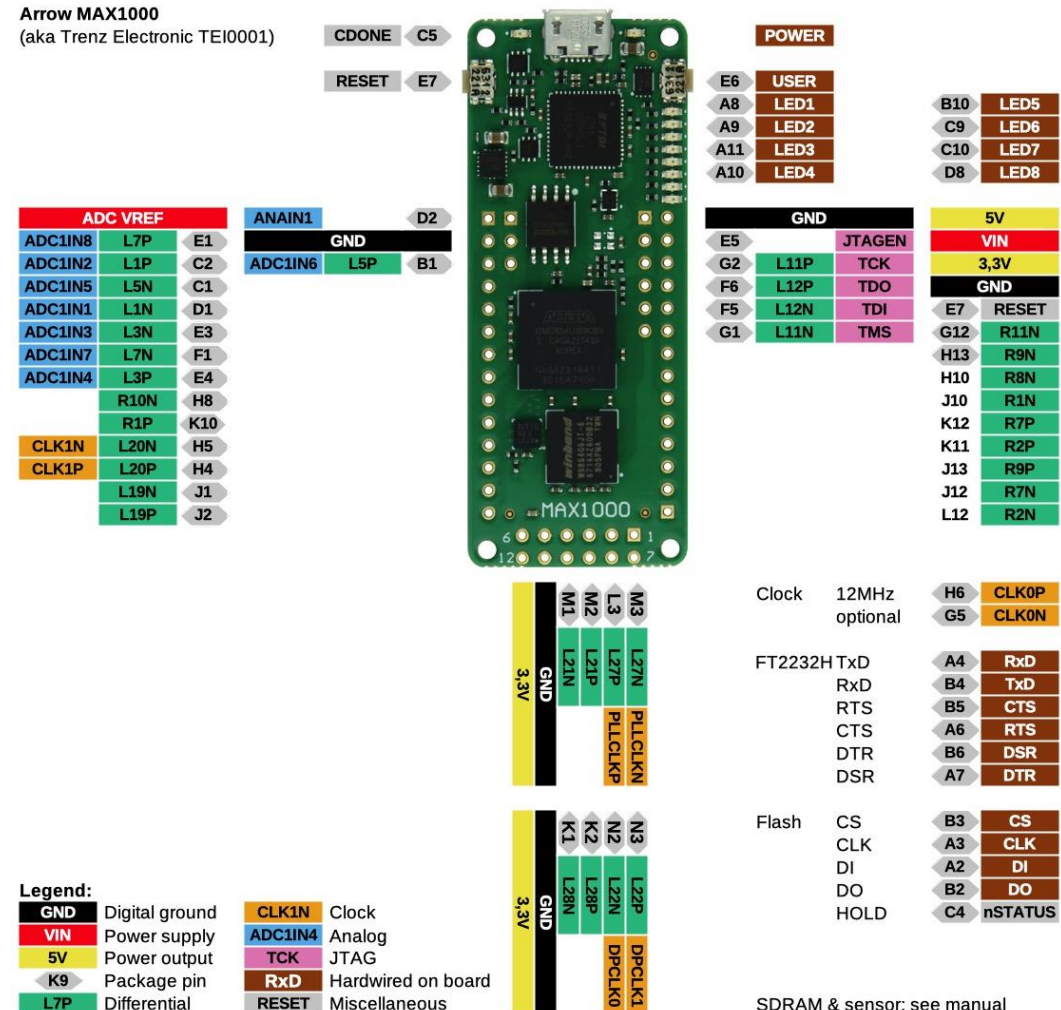




# Max1000 Pin Out

Pmod is master – Important as 3v3 is output not input – Be careful if connecting to another Pmod Interface.

Arrow MAX1000  
(aka Trenz Electronic TEI0001)



# The STM32F723 Disco

Contains Arm Cortex M7 processor with FPU.  
Very flexible protocol support with USB, CAN, SPI, I2C, I2S and UART

## Connectivity Provides

- Arduino Shield
- Pmod
- Touch Screen
- Wifi
- On board St-Link and USB UART

<b>System</b> Power supply 1.2 V regulator POR/PDR/PVD Xtal oscillators 32 kHz + 4 ~26 MHz Internal RC oscillators 32 kHz + 16 MHz PLL Clock control RTC/AWU 1x SysTick timer 2x watchdogs (independent and window) 79/112/138 I/Os Cyclic redundancy check (CRC)	<b>ART Accelerator™</b>  Cache I/D 8+8 Kbytes  <b>ARM Cortex-M7</b> 216 MHz	512-Kbyte Flash memory 256-Kbyte SRAM + 16-Kbyte ITCM RAM FMC/SRAM/NOR/NAND/SDRAM Dual Quad-SPI 1024-byte + 4-Kbyte backup SRAM 528-byte OTP
		<b>Connectivity</b> 5x SPI, 3x I²S, 3x I²C 1x CAN 2.0B 1x USB 2.0 OTG FS/HS USB HS Phy 1x USB 2.0 OTG FS 2x SDMMC 4x USART + 4 UART LIN, smartcard, IrDA, modem control 2x SAI (Serial audio interface)
<b>Control</b> 2x 16-bit motor control PWM synchronized AC timer 10x 16-bit timers 2x 32-bit timers LP timer	Floating point unit (FPU) Nested vector interrupt controller (NVIC) JTAG/SW debug/ETM Memory Protection Unit (MPU) PC-ROP	<b>Analog</b> 2x 12-bit, 2-channel DACs 3x 12-bit ADC 24 channels / 2.4 MSPS Temperature sensor
	AXI and Multi-AHB bus matrix 16-channel DMA True random number generator (RNG)	



# I2C or SPI

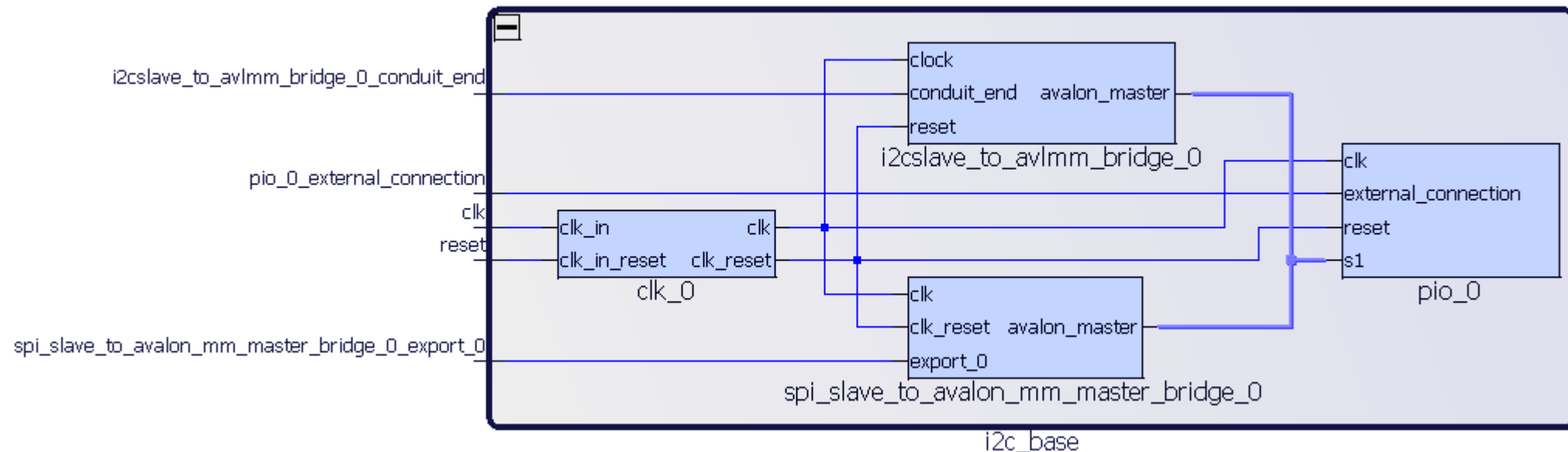
- Of course, depends on the interfaces available on the processor
- Both commonly provided in embedded systems
- SPI more complex and requires more lines
- I2C simpler – only two lines
- SPI higher performance – Full duplex
- I2C – Half Duplex
- SPI – Expansions for Quad and Octal to achieve much higher data rates
- I2C – Multi Master Support – Multiple Masters could access the Companion Chips

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00:				--	--	--	--	--	--	--	--	--	--	--	--	--
10:	--	--														
20:	--	--														
30:	--	--														
40:	--	--														
50:	--	--	--	--	--	55	--	--	--	--	--	--	--	--	--	--
60:	--	--														
70:	--	--														

# Solution Architecture

Architecture of the design provides multiple Avalon Master access points.

I2C and SPI slave can access the Avalon bus (via Interconnect) to access the implemented slave peripherals



Note, SPI and I2C in same design is for reference only

# Quartus Tool Chain

Design Created in Quartus, targeting Intel Max 10 device.

Development utilized Platform Designer – System Integration tool.

Debugging on target can be achieved using Signal Tap – In chip Logic Analyzer.

New Project Wizard

### Family, Device & Board Settings

Device Board

Select the family and device you want to target for compilation.  
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family

Family: MAX 10 (DA/DF/DC/SA/SC)

Device: All

Target device

☐ Auto device selected by the Fitter

☒ Specific device selected in 'Available devices' list

☐ Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: 169

Core speed grade: Any

Name filter: sau

☒ Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit elements	PLLs	Global Clocks	Max
10M04SAU169C8G	3.3V	4032	130	130	193536	40	1	10	242483
10M04SAU169I7G	3.3V	4032	130	130	193536	40	1	10	242483
10M08SAU169A7G	3.3V	8064	130	130	387072	48	1	10	255590
10M08SAU169C8G	3.3V	8064	130	130	387072	48	1	10	255590
10M08SAU169C8GES	3.3V	8064	130	130	387072	48	1	10	255590
10M08SAU169I7G	3.3V	8064	130	130	387072	48	1	10	255590
10M08SAU169I7P	3.3V	8064	130	130	387072	48	1	10	255590
10M16SAU169C8G	3.3V	15840	130	130	562176	90	1	20	458752
10M16SAU169I7G	3.3V	15840	130	130	562176	90	1	20	458752

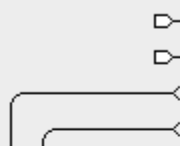


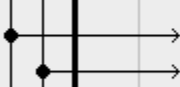

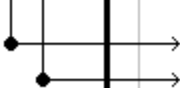

< >

< Back Next > Finish Cancel Help

# Platform Designer

Platform designer allows us to leverage existing IP

Custom IP can be created for easy integration to the Avalon MM bus

Use	Connections	Name	Description	Export	Clock	Base
<input checked="" type="checkbox"/>		<b>clk_0</b> clk_in clk_in_reset clk clk_reset	Clock Source Clock Input Reset Input Clock Output Reset Output	clk reset <i>Double-click to export</i> <i>Double-click to export</i>	<i>exported</i> clk_0	
<input checked="" type="checkbox"/>		<b>i2cslave_to_avlmm...</b> clock	I2C Slave To Avalon-MM Master Bridg... Clock Input	<i>Double-click to export</i>	clk_0	
		<b>avalon_master</b> reset conduit_end	Avalon Memory Mapped Master Reset Input Conduit	<i>Double-click to export</i> <i>Double-click to export</i> <b>i2cslave_to_avlmm_bridge_0_conduit_end</b>	[clock] [clock]	
<input checked="" type="checkbox"/>		<b>pio_0</b> clk reset	PIO (Parallel I/O) Intel FPGA IP Clock Input Reset Input	<i>Double-click to export</i> <i>Double-click to export</i>	clk_0 [clk]	
		<b>s1</b> external_connection	Avalon Memory Mapped Slave Conduit	<i>Double-click to export</i> <b>pio_0_external_connection</b>	[clk]	0x0000_0000
<input checked="" type="checkbox"/>		<b>spi_slave_to_avalon...</b> clk clk_reset export_0	SPI Slave to Avalon Master Bridge Int... Clock Input Reset Input Conduit	<i>Double-click to export</i> <i>Double-click to export</i> <b>spi_slave_to_avalon_mm_master_bridge_0_export_0</b>	clk_0 [clk]	
		<b>avalon_master</b>	Avalon Memory Mapped Master	<i>Double-click to export</i>	[clk]	

# Address Space

Address space can be optimized for the size of the peripheral being driven - no need to select 32 bit when only 8 bit addressing needed.

Selecting correct, address width will ensure the higher bandwidth as the transactions are more efficient.

System Contents	Address Map	Interconnect Requirements	Schematic
<b>System:</b> i2c_base			
	i2cslave_to_avlmm_bridge_0.avalon_master	spi_slave_to_avalon_mm_master_bridge_0.avalon_master	
pio_0.s1	0x0000_0000 - 0x0000_000f	0x0000_0000 - 0x0000_000f	



# Accessing Avalon MM over I2C & SPI

Explain read and write example – for both SPI and I2C

Software access at lowest level

Bare Metal – Hardware abstraction libraries

With Linux we can leverage frameworks like i2c-tools, Smbus and I2C and SPI Dev

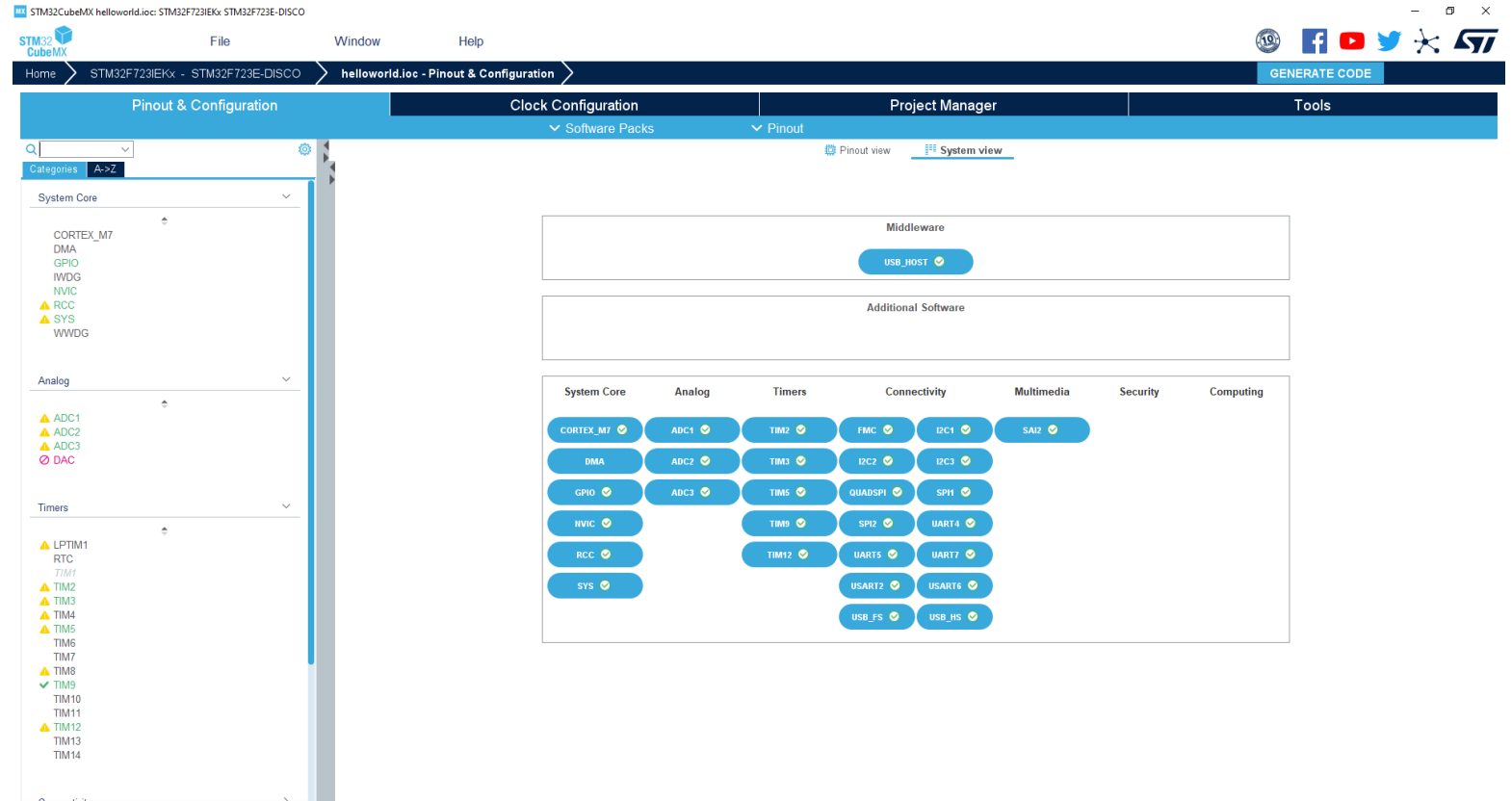
Make it very easy for Linux systems to be able to interact with the companion board

# STM Software Solution

To create software solution for STM32F723 Discovery we need the following

STM Cube MX - Generates the low-level system configuration and start up code along with the project.

STM Cube IDE - IDE used for development of the application SW



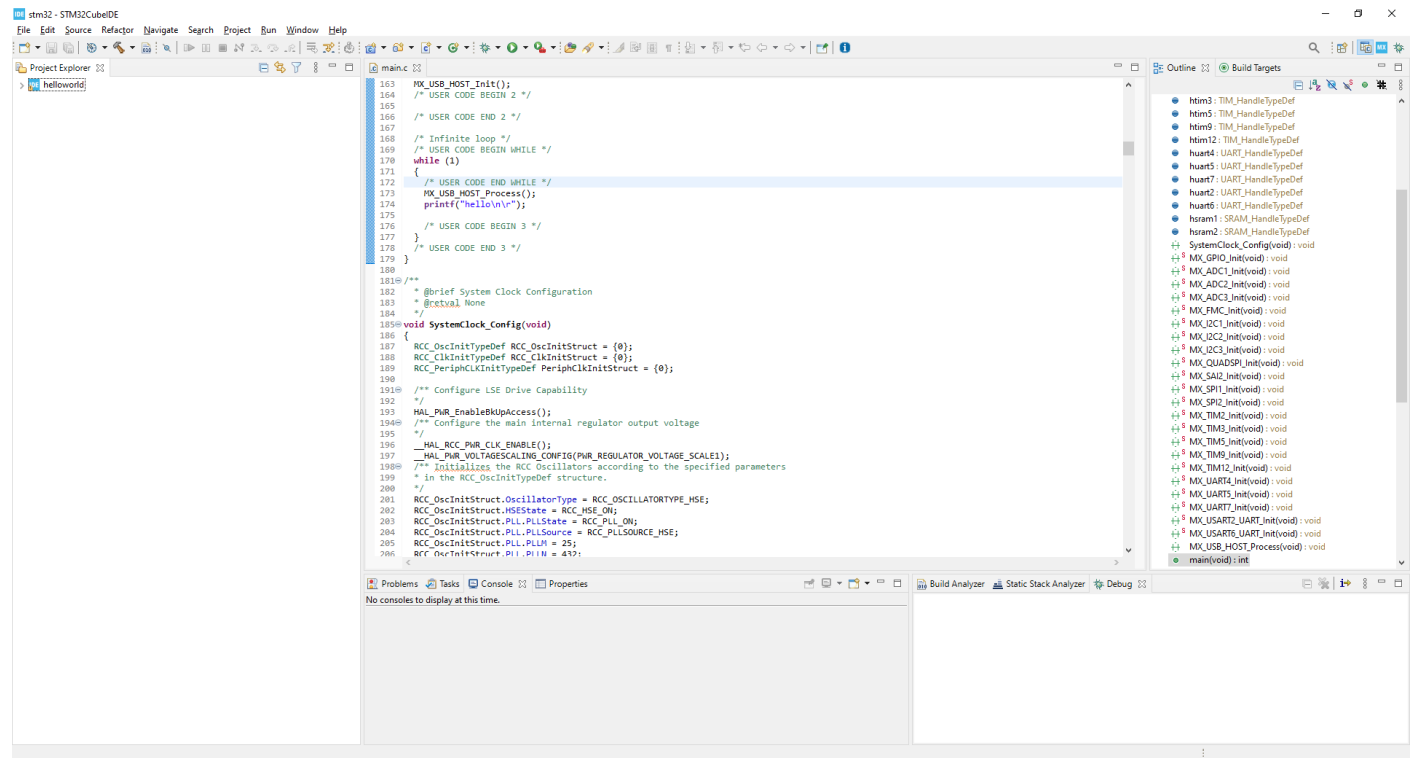
# Software Development

Application development uses C.

Access to Hardware via Hardware Abstraction Layer (HLA)

Debugging via ST-Link and GDB.

- » Single Step / Step Into
- » Breakpoints
- » Watch Values
- » Memory View etc



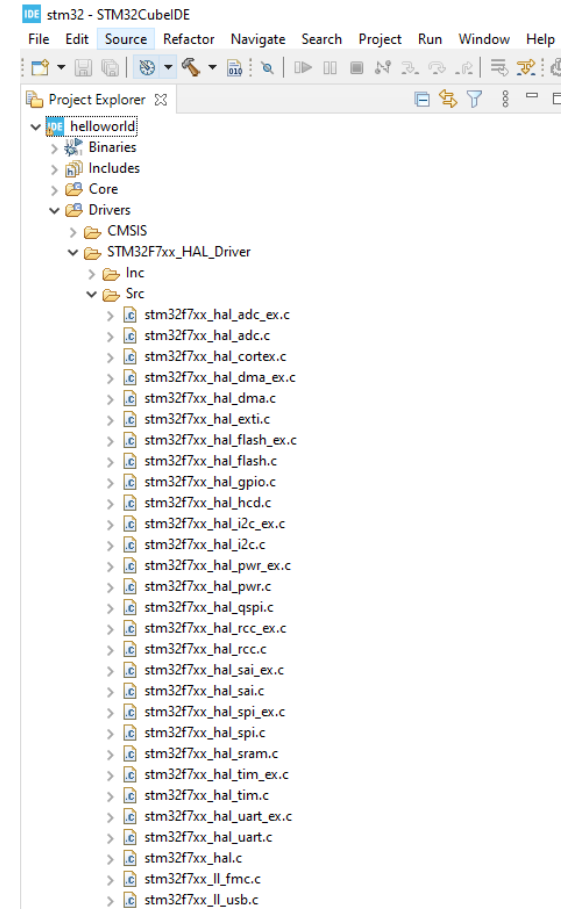
# Hardware Abstraction Layer

```

619         uint16_t Size);
620 HAL_StatusTypeDef HAL_I2C_Master_Receive_IT(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData,
621         uint16_t Size);
622 HAL_StatusTypeDef HAL_I2C_Slave_Transmit_IT(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size);
623 HAL_StatusTypeDef HAL_I2C_Slave_Receive_IT(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size);
624 HAL_StatusTypeDef HAL_I2C_Mem_Write_IT(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress,
625         uint16_t MemAddSize, uint8_t *pData, uint16_t Size);
626 HAL_StatusTypeDef HAL_I2C_Mem_Read_IT(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress,
627         uint16_t MemAddSize, uint8_t *pData, uint16_t Size);
628
629 HAL_StatusTypeDef HAL_I2C_Master_Seq_Transmit_IT(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData,
630         uint16_t Size, uint32_t XferOptions);
631 HAL_StatusTypeDef HAL_I2C_Master_Seq_Receive_IT(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData,
632         uint16_t Size, uint32_t XferOptions);
633 HAL_StatusTypeDef HAL_I2C_Slave_Seq_Transmit_IT(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size,
634         uint32_t XferOptions);
635 HAL_StatusTypeDef HAL_I2C_Slave_Seq_Receive_IT(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size,
636         uint32_t XferOptions);
637 HAL_StatusTypeDef HAL_I2C_EnableListen_IT(I2C_HandleTypeDef *hi2c);
638 HAL_StatusTypeDef HAL_I2C_DisableListen_IT(I2C_HandleTypeDef *hi2c);
639 HAL_StatusTypeDef HAL_I2C_Master_Abort_IT(I2C_HandleTypeDef *hi2c, uint16_t DevAddress);
640
641 /***** Non-Blocking mode: DMA */
642 HAL_StatusTypeDef HAL_I2C_Master_Transmit_DMA(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData,
643         uint16_t Size);
644 HAL_StatusTypeDef HAL_I2C_Master_Receive_DMA(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData,
645         uint16_t Size);
646 HAL_StatusTypeDef HAL_I2C_Slave_Transmit_DMA(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size);
647 HAL_StatusTypeDef HAL_I2C_Slave_Receive_DMA(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size);
648 HAL_StatusTypeDef HAL_I2C_Mem_Write_DMA(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress,
649         uint16_t MemAddSize, uint8_t *pData, uint16_t Size);
650 HAL_StatusTypeDef HAL_I2C_Mem_Read_DMA(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress,
651         uint16_t MemAddSize, uint8_t *pData, uint16_t Size);
652
653 HAL_StatusTypeDef HAL_I2C_Master_Seq_Transmit_DMA(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData,
654         uint16_t Size, uint32_t XferOptions);
655 HAL_StatusTypeDef HAL_I2C_Master_Seq_Receive_DMA(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData,
656         uint16_t Size, uint32_t XferOptions);
657 HAL_StatusTypeDef HAL_I2C_Slave_Seq_Transmit_DMA(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size,
658         uint32_t XferOptions);
659 HAL_StatusTypeDef HAL_I2C_Slave_Seq_Receive_DMA(I2C_HandleTypeDef *hi2c, uint8_t *pData, uint16_t Size,
660         uint32_t XferOptions);
661
662 /**
663  * @}

```

Number of function calls which can be accessed from the main application to implement I2C or SPI transactions.





# Demonstration



# Wrap Up

Demonstrated a flexible method for creating a companion chip design expanding IO or supporting bespoke or new interfaces.

Demonstration outlined how to use I2C to break out to LED GPIO using the MAX1000 board - Can be expanded to support a range of IO

Creation of project for the Intel FPGA can be achieved in 30-60 minutes, with similar for SW development of the simple application software.

What do you want to do with your application ?

# Call to Action

All instructions and files to recreate the project can be found at  
[https://github.com/ATaylorCEngFIET/Intel\\_Max1000](https://github.com/ATaylorCEngFIET/Intel_Max1000)

MAX1000:

<https://www.arrow.com/en/products/max1000/arrow-development-tools>

STM32 Discovery Kit:

<https://www.arrow.com/en/products/stm32f723e-disco/stmicroelectronics?q=STM32F723E-Disco>



# Questions ?

Something comes to mind after, the session email [adam@adiuvoengineering.com](mailto:adam@adiuvoengineering.com)



# ADIUVO

ENGINEERING AND TRAINING, LTD.

[www.adiuvoengineering.com](http://www.adiuvoengineering.com)



[adam@adiuvoengineering.com](mailto:adam@adiuvoengineering.com)