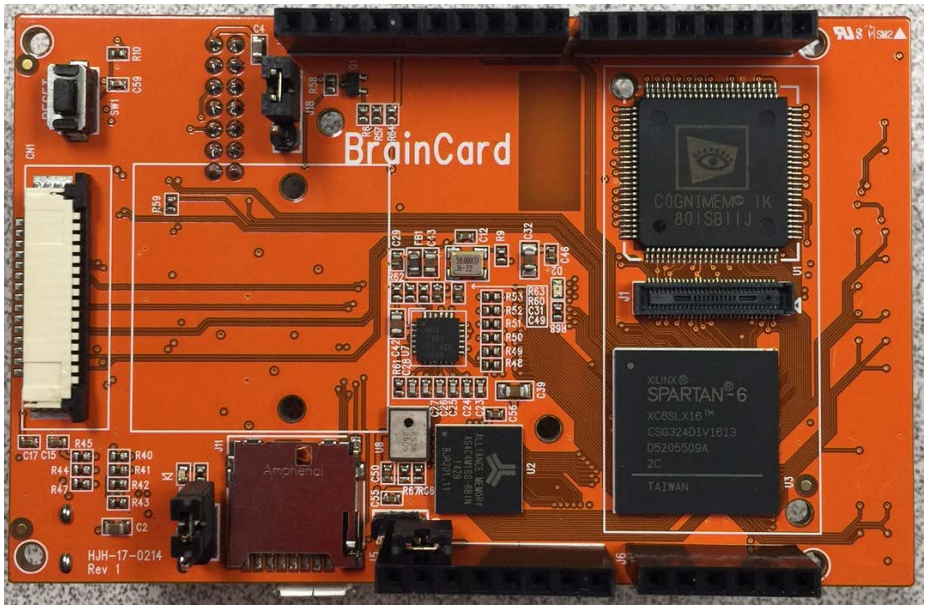
BrainCard,
Low-power, trainable pattern recognition for IoT



Version 1.6

Revised 11/07/2017

GENERALVISION

Contents

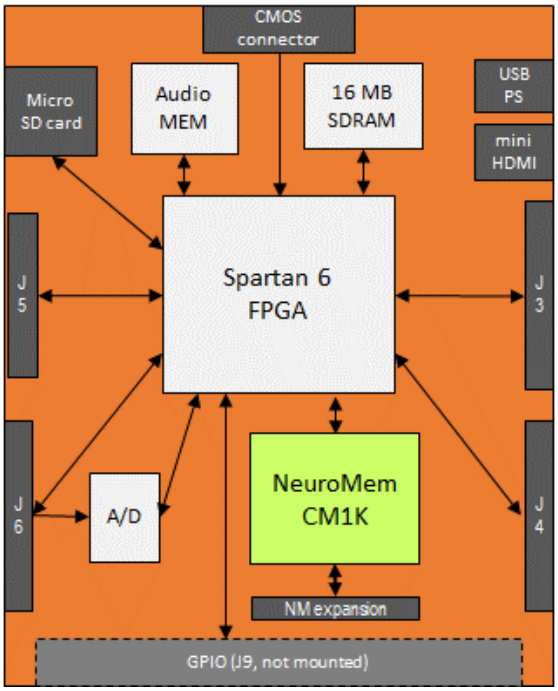
BrainCard is a product of General Vision, Inc. (GV) manufactured by nepes (Korea)

1 Introduction

BrainCard is a trainable pattern recognition board for IoT and smart appliances featuring the NeuroMem CM1K chip with 1024 neurons and a Field Programmable Gate Array which can interface the neurons and the wealth of components mounted on the BrainCard to an Arduino or Raspberry PI microcontroller.

Sensor data collected through Arduino shields and other plug-in modules can be assembled into feature vectors of up to 256 bytes. Their learning can be triggered by external user inputs, programmable time stamps, but also by the detection of novelties made by the neurons themselves. Once trained, the neurons can be continuously interrogated to report known patterns or events, or to report novelty. Depending on your application, the output of the neurons can control actuators, trigger a selective recording or transmission or else. Applications include identification, surveillance, tracking, adaptive control and more.

The default firmware features a simple communication protocol between your microcontroller and the NeuroMem CM1K chip. This enables the use of the NeuroMem neurons to learn and recognize patterns generated by or collected through your Arduino, Raspberry PI, STM32Nucleo and other microcontroller.



	Default	Ideas for additional features. Refer to Chapter 7
Access to neurons & expansion	YES	n/a
Run-Time engine	NO	Described later in this manual.
SDRAM	NO	Store data acquired directly on the board or transmitted by microcontroller
Micro SD card	File Read access reserved for the FPGA reconfiguration. Described later in this manual.	Save and Restore knowledge built and stored in the neurons.
CMOS	NO	Acquire video for immediate recognition by the engine programmed in the FPGA or intermediary storage to SDRAM
Audio MEM	NO	Acquire sound for immediate recognition by the engine programmed in the FPGA or intermediary storage to SDRAM
A/D	NO	Acquire signal for immediate recognition by the engine programmed in the FPGA or intermediary storage to SDRAM
Mini HDMI	NO	Miscellaneous display
GPIO	Power supply and SPI bus of the Raspberry PI	Miscellaneous actuation

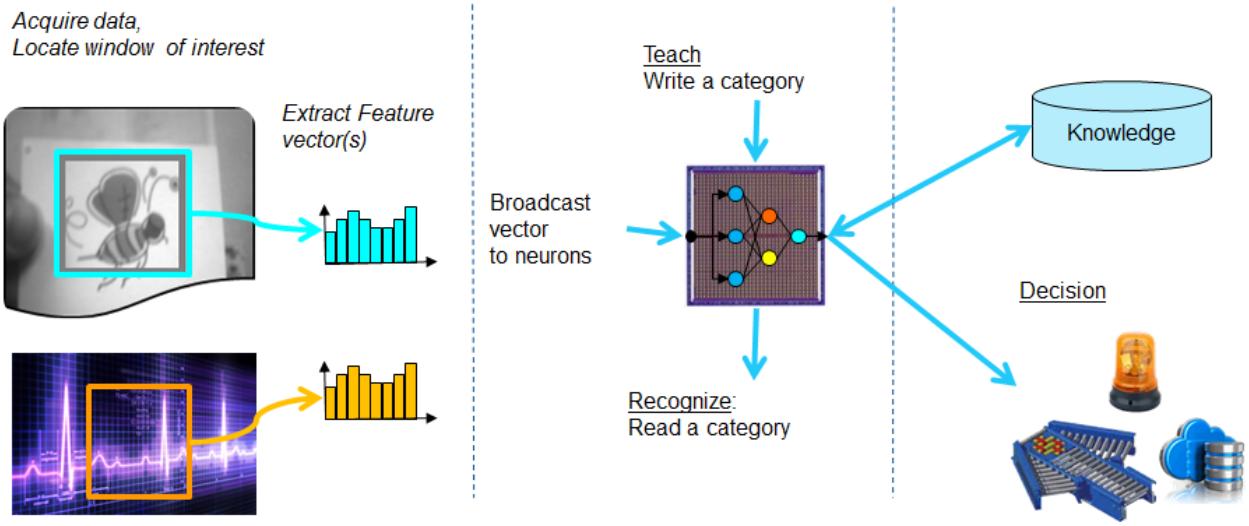
The NeuroMem network

The NeuroMem CM1K chip of the BrainCard is very easy to use and access from an Arduino, Raspberry PI, STM32Nucleo and other microcontrollers with an SPI protocol compliant to the firmware of the BrainCard.

Step 1
Microcontroller
Data coming from sensors must be converted into pattern vectors of up to 256 bytes.The pattern vectors are then broadcasted to the NeuroMem neurons for either learning or recognition. All you must do is focus on the quality of the input signals, the selection of relevant and discriminant feature extractions.

Step 2
BrainCard_Neurons library
The NeuroMem neurons handle the automatic learning of your examples and associated categories, the recognition of new patterns, the detection of uncertainty cases if any, the detection of drifts or decrease of confidence, the reporting of novelties and/or anomalies.

Step 3
Microcontroller
Finally, you must format the response of the neurons to convert it into a global decision or action for your application, without forgetting to save the knowledge built by the neurons for backup purposes or distribution to other systems.



2 Getting started

Go to http://www.general-vision.com/BrainCardBSP and retrieve the Board Support Package which contains libraries and examples for various programming IDE.

Power supply

The Braincard is an Arduino Shield board powered by default through the Arduino board. The power can then be delivered through the Arduino USB port or its external power supply port.

have the choice to use the through the USB port (default), or the Arduino connector (J5)
Selection of one or the other is made through the jumper J8:

- Pin 1-2: Power through the Arduino board and delivered on connector J5 of the Braincard
- Pin 2-3: Power through the USB connector on the back panel of the BrainCard

In term of IOs, the BrainCard only accepts 3.3V I/O on the Arduino connectors J 3 and J4.

Hardware compatibility

Communication with the BrainCard is made through via SPI through pins D10-D13 on the J3 jumper. Its SPI protocol described in this manual and already implemented in the Arduino and Raspberry PI examples supplied in the Board Support Package.

We have tested the BrainCard as a shield with the following boards:

- Arduino UNO
- ADAFruit Metro
- Intel Arduino/Genuino 101
- Kocoafab Orange board
- STM32nucleo
- Raspberry PI 2

Depending on the base board, you may have to use spacers or elbow connectors to ensure that the BrainCard is properly plugged in and powered.

BrainCard shield for Arduino101

BrainCard shield for Raspberry PI

Do not forget to set SPI enabled under the Raspberry PI configuration panel.

BrainCard API

Example programs are supplied for the Arduino IDE and also in Python for the Raspberry Pi. Detailed comments are included in the source code and print statements executed by the scripts.

The functions of the NeuroMem library are described in the technical manual TM_NeuroMem_API.pdf and divided as follows:

NeuroMemSPI library

```
NeuroMemSPI();
int connect(int Platform);
int FPGArev();
int read(unsigned char reg);
void write(unsigned char reg, int data);
void writeAddr(long addr, int length, int data[]);
void readAddr(long addr, int length, int data[]);
```

NeuroMemAI library

```
NeuroMemAI();
int begin(int Platform);
void forget();
void forget(int Maxif);
void clearNeurons();
int countNeuronsAvailable();

void setContext(int context, int minif, int maxif);
void getContext(int* context, int* minif, int* maxif);
void setRBF();
void setKNN();

int broadcast(unsigned char vector[], int length);
int learn(unsigned char vector[], int length, int category);
int classify(unsigned char vector[], int length);
int classify(unsigned char vector[], int length, int* distance, int* category, int* nid);
int classify(unsigned char vector[], int length, int K, int distance[], int category[], int nid[]);

void readNeuron(int nid, unsigned char model[], int* context, int* aif, int* category);
void readNeuron(int nid, unsigned char neuron[]);
int readNeurons(unsigned char neurons[]);
int writeNeurons(unsigned char neurons[]);
```

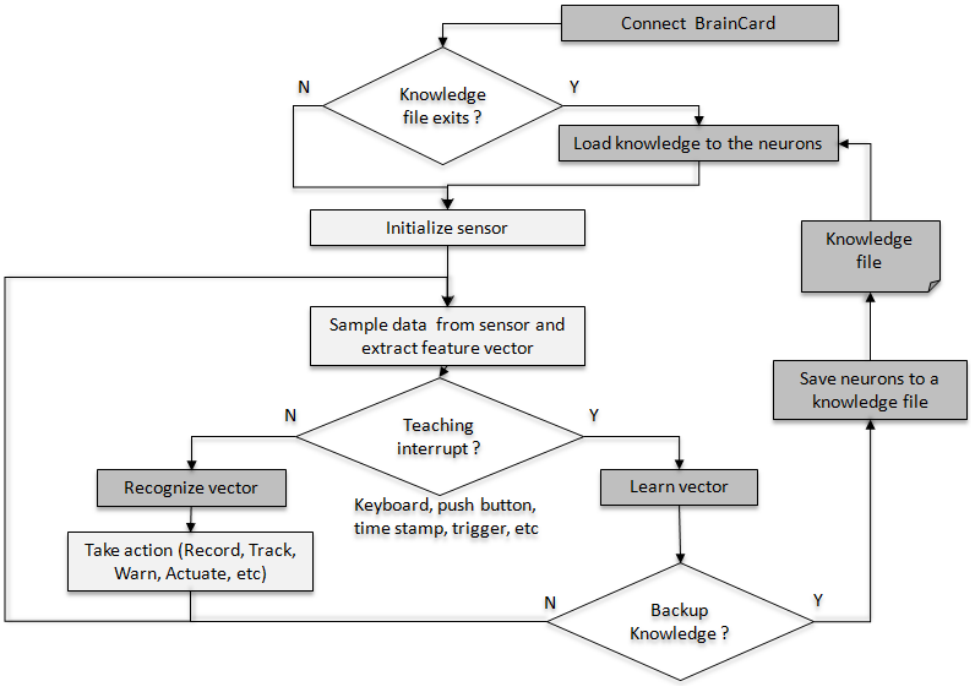
Installation on Arduino IDE:

Copy the NeuroMem folder (or former BrainCardNeurons folder) to MyDocuments\\Arduino\\libraries
Launch the Arduino IDE
Under Tools/Board Manager menu, select your Arduino processor board (Uno, 101, STM32, etc.)
Under File/Examples/NeuroMem, you can access

- generic examples for all NeuroMem-Smart devices
- hardware specific examples (using the Arduino101 featuring a IMU sensor hub)

3 Typical application workflow

The data collected through the sensors can be broadcasted to the neurons for learning and recognition. Learning can be triggered by external user inputs, time stamps, but also the ability of the neurons to detect drifts from learned models. Learning can also be done “off line” on data previously collected and saved. Recognition can consist of using the neurons to classify input patterns or identifying novel or abnormal patterns. Depending on your application, the output of the neurons can control actuators, trigger a selective recording or transmission or else. Applications include identification, surveillance, tracking, adaptive control and more.



How do the neurons learn and recognize?

Our NeuroMem RBF tutorial is a simple application demonstrating how the neurons build a decision space autonomously by learning examples and how they recognize new vectors with ability to report cases of unknown and uncertainties too. For the sake of a simplicity, the decision space is limited to a 2D space but the mechanism is the same to recognize a (X1, X2) vector as well as an (X1, X2, … X127) vector which is the maximum length supported by the neurons of the CM1K.

For more information of the NeuroMem technology, please refer to the NeuroMem Technology Reference Guide.

What is a feature vector?

The neurons are agnostic to the data source which means that they ready to learn and recognize feature vectors extracted from text (parsing), discrete measurements, audio signal, video signal, digital images, etc. The only constraint is that the vector must be formatted as a byte array of maximum length 256.

In the case of images, a feature vector can be a subsampling of pixels within a region of interest, a histogram, some histogram of gradients within a patch of pixels, etc. In the case of an audio or biosensor signal, the feature vector can be a set of signal samples taken at a specific sampling rate, a histogram of peaks or zero crossing, or a power spectrum, etc.

4 Academic examples for all hardware platforms

Test_SimpleScript

Simple script stimulating the neurons to learn and recognize patterns generated programmatically. Detailed description of this script is available at http://www.general-vision.com/documentation/TM_TestNeurons_SimpleScript.pdf.

Test_SimpleScript2

This script is similar to the Test_SimpleScript but also demonstrates additional capabilities of the neurons which are accessible in the BrainCardNeurons library Pro, including but not limited to the use of the KNN, the use of the LSup/L1 norms for the calculation of distances, and the use of multiple contexts to learn and classify vectors representing different dimensions or data types within the same NN.

In addition to the script described in http://www.general-vision.com/documentation/TM_TestNeurons_SimpleScript.pdf , you can practice with:

- Switching the NN from Radial Basis Function mode to K-Nearest Neighbor mode and comparing results between the two classifiers using a same knowledge and same input vector. KNN always gives a response for the closest match while RBF can report cases of "unknown" and cases of uncertainties.
- Training the neurons with vectors belonging to 3 different contexts with the 3rd context activating the Lsup norm.

For more information about the RBF/KNN classifiers, the definition of Contexts, the use of different Norms, please refer to the NeuroMem Technology Reference Guide.

5 Examples using the MEMS of the Arduino 101

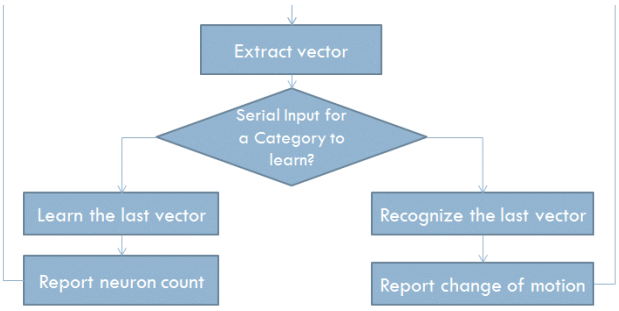
Test_Neuron_andIMU1

This script requires to plug the BrainCard on an Arduino/Genuino101. It uses 6-axis IMU of the Intel Curie module and the neurons of the CM1K.

The script assembles signals from the accelerometer and gyroscope into a simple feature vector broadcasted continuously to the neurons for recognition. Learning is performed by entering a category value through the serial input. Refer to the General Vision movie tutorial.

In the Arduino code, the LOOP continuously reads the IMU signals and extracts a simple feature vector which is a normalized subsampling of the 6 axes. The recognition of this vector starts automatically as soon as the neurons have some knowledge. The neurons build the knowledge as soon as you start teaching examples.
When you enter a category through the serial port, the program associates it to the last feature vector. When teaching a vertical motion for example, you want to teach more than once to make sure the neurons learn an example of a vertical up-to-down and vertical down-to-up, obviously at selected speed and amplitudes.

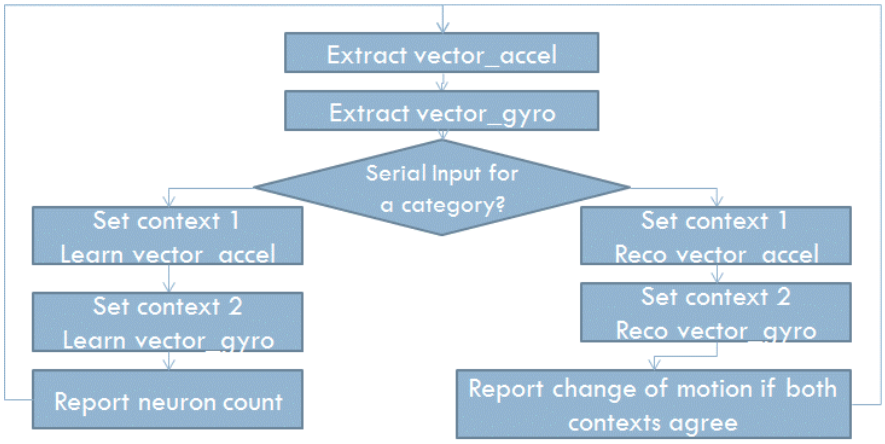
The script mentions 2 categories (vertical and horizontal) for the sake of simplicity, but feel free to define your own categories combining not only a direction, but also other motion such as circular, a range of amplitude and speed, etc. Remember that the category value can range between 0 to 32767 which means that it can encode multiple criteria (ex: bit[2:0] for direction, bit [5:3] speed range, etc.).



Test_Neuron_andIMU2

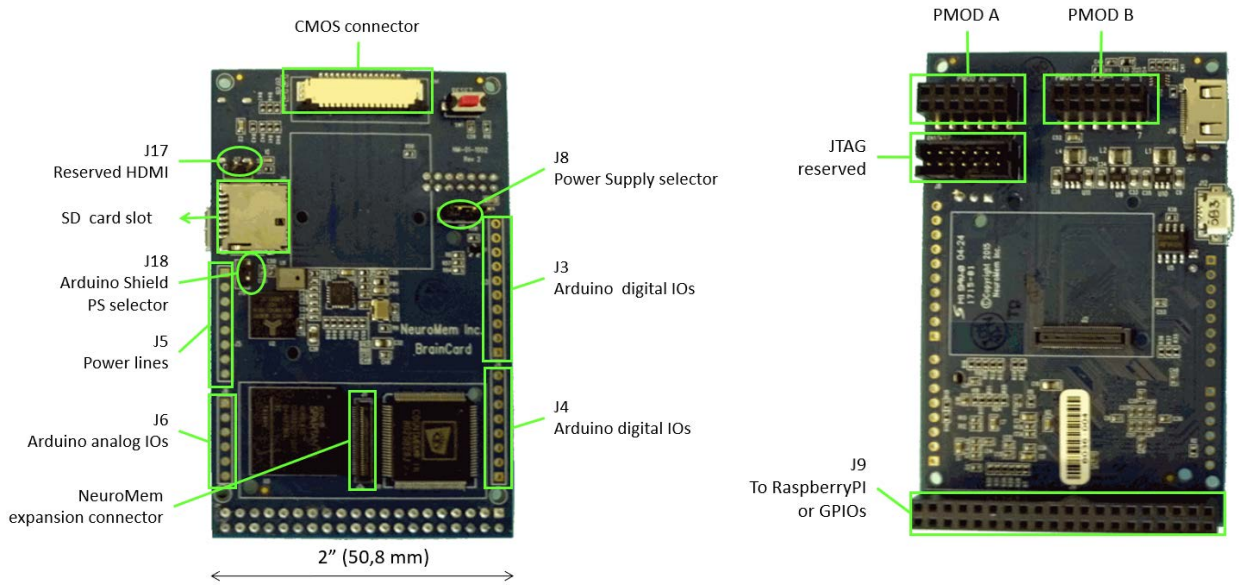
Test_Neurons_andIMU2 is similar to the Test_Neurons_andIMU1 and illustrates the ability of the NeuroMem neurons to handle multiple networks in a same chip for more robust decision making. This script requires to plug the BrainCard on an Arduino/Genuino101. It uses 6-axis IMU of the Intel Curie module and the neurons of the CM1K.

The signals of the accelerometers and gyroscope are assembled into two separate feature vectors and associated to 2 different contexts. Learning a motion builds 2 decision spaces at once and consequently commits more neurons. The script displays a positive response only if both sub-networks agree with the classification, thus producing a more conservative but accurate response than in the script Test_Neurons_andIMU.



Remark1: This example is very academic and assemble a pattern which should be more sophisticated for real-life system taking a calibration into account, integrating a sampling rate adequate for the type of motion and profiling the waveforms more selectively using distances between peaks and zero crossing, etc.

6 Connectors



NOTE: The J9 connector and the 2 PMOD connectors are no longer mounted on the new version of the BrainCard.

J1 NeuroMem expansion connector

J8 **Power source selection**

- Pin 1-2: Power through the Arduino board and delivered on connector J5 of the Braincard
- Pin 2-3: Power through the USB connector on the back panel of the BrainCard

J17 Optional 5 Volts power supply to HDMI through the BrainCard. Default is OFF (no jumper).

J19 Optional 3.3 Volts power supply to an Arduino Shield if the 3.3V pin of the Arduino power connector is not already powered. This feature can be useful if the BrainCard is interfaced to an Intel Edison module or Raspberry PI.

J5

Pins	Signal
1	NC
2	NC
3	NC
4	NC/ 3.3V**
5	5V
6	GND
7	GND
8	Reserved

J3

Pins	Signal (3.3V max)
1	D8 / FLASH_CS8
2	D9 / SD_CS9
3	D10 / SPI_CS
4	D11 /MOSI
5	D12 / MISO
6	D13 / SPI_CLK
7	NC
8	AREF
9	SDA
10	SCL

J4

Pins	Signal (3.3V max)
1	D0 (alt RX)
2	D1 (alt TX)
3	D2
4	D3
5	D4
6	D5
7	D6
8	D7

J6

Pins	Signal
1	A0
2	A1
3	A2
4	A3
5	A4
6	A5

** disconnected by default. See J19.

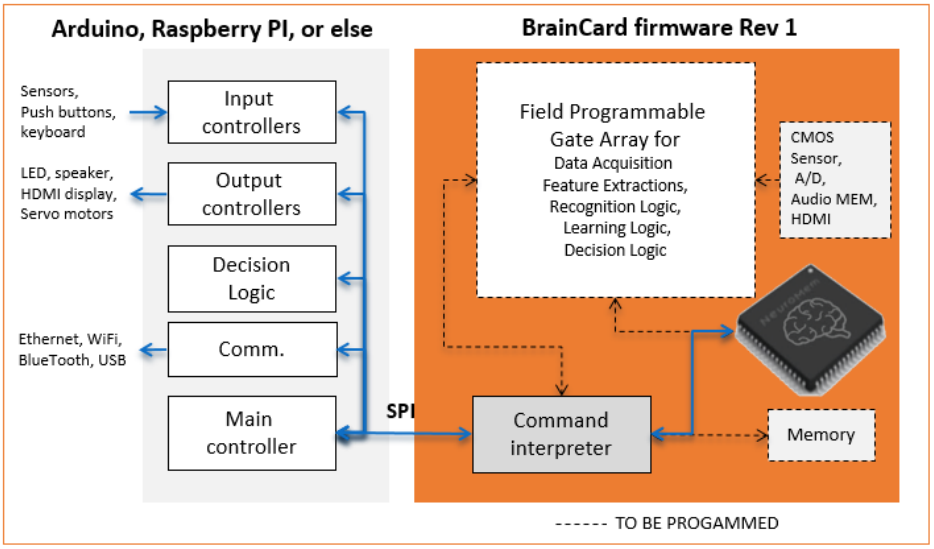
Raspberry Pi connector (J9)

Pins J9	Signal	Pins	Signal
1	NC	2	5 Volts In
3	I2C1SDA	4	5 Volts In
5	I2C1SCL	6	GND
7	GPIO4	8	UART_TXD
9	GND	10	UART_RXD
11	GPIO17	12	GPIO18
13	GPIO27	14	GND
15	GPIO22	16	GPIO23
17	NC	18	GPIO24
19	MOSI	20	GND
21	MISO	22	GPIO25
23	SCLK	24	GPIO8
25	GND	26	GPIO7
27	NC	28	NC
29	GPIO5	30	GND
31	GPIO6	32	GPIO12
33	GPIO13	34	GND
35	GPIO19	36	GPIO16
37	GPIO26	38	GPIO20
39	NC	40	GPIO21

7 BrainCard FPGA Configuration

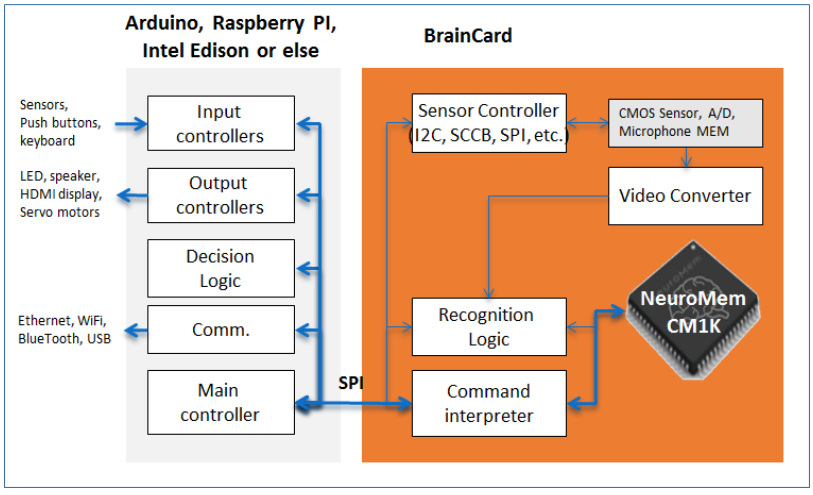
FPGA default configuration

Our initial firmware version 1 is limited to a simple communication protocol between your microcontroller and the CM1K. This enables the use of the NeuroMem neurons to learn and recognize patterns generated by or collected through a shield by your microcontroller. Examples supplied for the Arduino IDE manipulate numeric data as well as data coming from accelerometers. Check http://general-vision.com/braincardbsp/ for links to latest videos and tutorials.



Programming idea for video surveillance

In addition to the neurons, the CM1K chip integrates a simple video recognition stage with direct digital video bus.



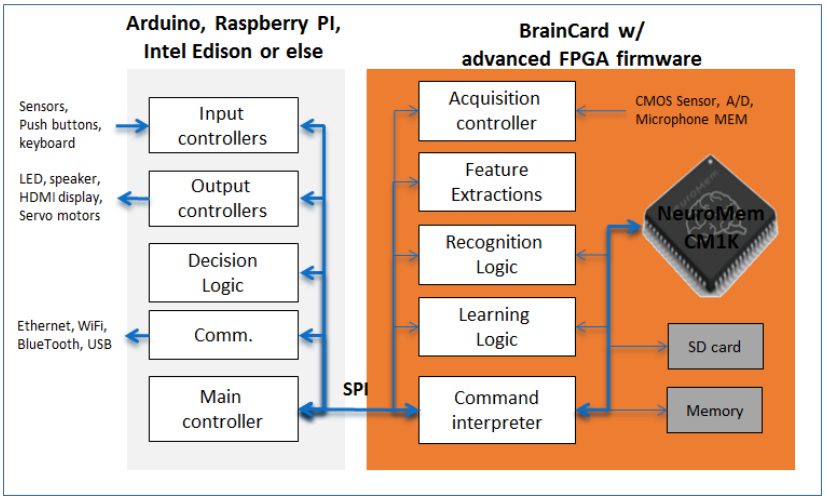
- A video converter can sample and convert the signal from the RaspiCam into 11 lines: V_CLK, L_Valid, F_Valid, V_Data (8). Refer to the CM1K manual for a description of the expected signals (chapter 4 and paragraph 6.4)
- Depending on the selected sensor, its settings such as Gain, Shutter, AGC, External Trigger, can be controlled through a serial protocol such as I2C, SPI, SCCB.
- The Recognition Logic simply connects the output of the Video Converter to the digital input bus of the 1st CM1K chip of the chain. The RECO_EN and VI_EN lines must also be pulled up in order for the recognition stage to run and bit 0 of the RSR register must be set to 1. Beyond this setup, the controller simply samples the DATA bus of the CM1K upon the rise of its CAT_VALID pulse and formats or pushes it As IS to the USB controller for output.

How to test that the video frame is properly captured?

- Option #1: Save the pixel values to MRAM and later retrieve the resulting byte array over USB for display as a 2D array with a size equal to frame width times frame height.
- Option #2: Set the Region Of Interest (ROI) of the CM1K to a 16x16 area. After each fall of the frame valid signal write the value k to the Category register, displace the ROI in a raster pattern with 16 pixels increment and increment k. When the ROI has reached the position of the last patch of 16x16 in the video frame, stop the continuous recognition by writing bit 0 of the RSR register to 0. From the PC host, execute the ReadNeurons function to retrieve the content of the neurons and re-assemble their models into an image.

Other Programming ideas for the BrainCard FPGA

Functionalities can be added into the FPGA firmware to reduce the number of I/O transfers between the microcontroller and the neurons as illustrated below. Once programmed in a new configuration file, updating the FPGA can be done very simply by loading this file to the Flash memory of the BrainCard, or by reprogramming the FPGA through its JTAG programmer. This procedure is described later in this manual.



Communication controller

Communication with the FPGA is presently established through an SPI protocol described in a next chapter. Functions performing the SPI_Read and SPI_Write between the BrainCard and the Arduino and Raspberry PI platforms are included in the BrainCard API.

Data acquisition

- Signal from A/D or microphone: Sample and store data to the Braincard memory
- Images: Acquire and store single or multiple frames to memory or SD card, control gain, shutter speed and trigger via I2C control command

Feature Extractions

- Signal: raw sampling, power spectrum, wavelet
- Images: color or monochrome subsamples, color and intensity histograms, SURFs, HOGs, etc.

Learning Logic

- Multi-modal (multiple sensor inputs sampled at the same time)
- Multi-scale, Multi-feature (extracted from the same input signal to build redundancy)

- Reinforced learning based on the recognition in previous samples

Recognition Logic

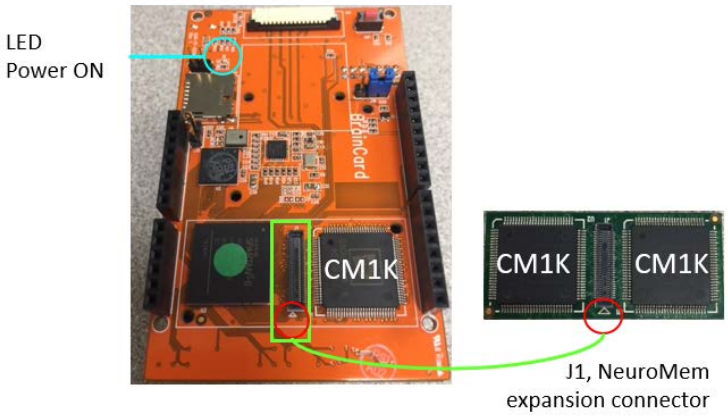
- Limited to the closest match
- Extended to the first N best match to build more robust decision
- Aggregate recognition derived from multiple sensors, multiple scales and multiple features at a same sampling time

Decision Logic

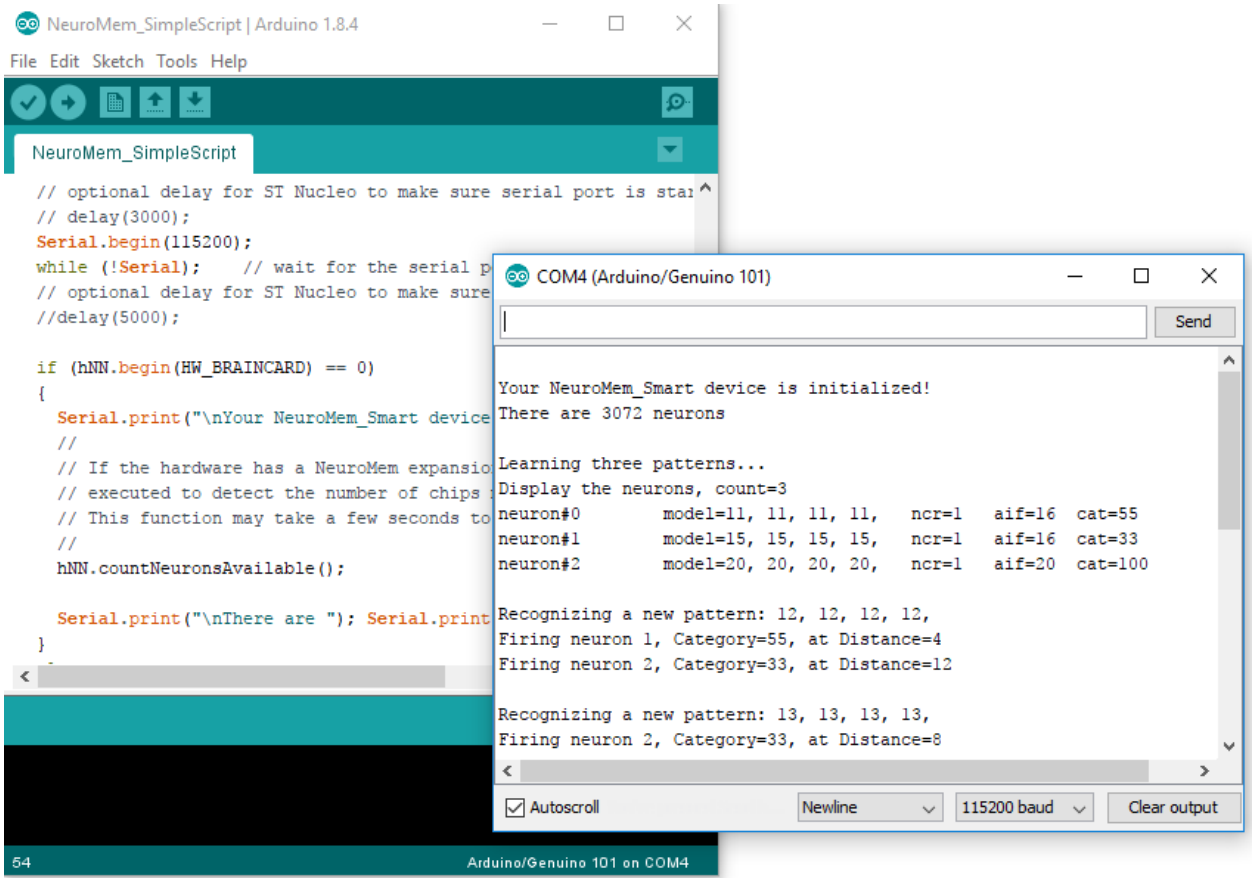
- Resizing and repositioning of the region or window of search based on the current recognition
- Switch sensor, switch feature extraction
- Rule based actuation/decision
- Pattern recognition based actuation/decision

8 Adding an Expansion module

- Before plugging the Expansion module, make sure to align its arrow with the arrow of the expansion connector (see diagram below)



- Verify that the blue LED is lit. If not, the USB power supply might be insufficient to power the 3 CM1K chips and you might want to use the external power supply of your Arduino board instead.
- Launch the NeuroMem_SimpleScript and make sure that the function hNN.countNeuronsAvailable() is executed in the Setup section of the program. It might be commented out in the default script since it is only useful to confirm the detection of expansion modules.
- Its execution should display the number of detected neurons as shown in the example below. In the case of one expansion with 2 CM1K chip, the total is 3027.



9 SPI Protocol

Communication to the BrainCard is established through J3 and follows the SPI protocol described below. SPI_Read and SPI_Write functions are included in the BrainCard API.

The SPI protocol is based on a 10-bytes control command described below and intended to access the various components of the board including the NeuroMem CM1K chip, the memory, SD card, A/D converter, CMOS sensor, and all.

The full memory map of the BrainCard will be published shortly, but is presently limited to access to the on-board CM1K chip or a chain of CM1K chips.

Address Range	**Module=** Address[30-24]	Functionality defined by registers = Address[23:8]
0x01000000 0x0100001F	NeuroMem CM1K	Access to the registers of the CM1K chip
TBD	Board settings	Access to the settings of the board
TBD	SDRAM	Access to the bank of SDRAM of the board connected to the host.
TBD	CMOS	Access to the camera module plugged on CM1
TBD	AD	Access to the A/D converter SPI lines
TBD	Microphone	Access to the microphone
TBD	SD	Access to the SD card
TBD	PMOD	Access to the PMOD A and B

Byte command sequence:

- Byte0 Reserved
- Byte 1-2-3-4 R/W bit + 31-bit address
- Byte 5-6-7 24-bit Data length expressed in number of words
- Byte 8-9 Minimum first 2 bytes of data
- Byte + Remaining data (up to Data Length)

Programming examples of the following functions are supplied in C++, Python and Arduino:

- Read_Addr(long **Address**, long **Length**, int[] **Data**)
- Write_Addr(long **Address**, long **Length**, int[] **Data**)
- Read(byte **ModuleID**, byte **Register**, int **Data**), subset of Read_Addr to read a single 16-bit register
- Write(byte **ModuleID**, byte **Register**, int **Data**), subset of Write_Addr to write a single 16-bit register

Write command

Reserved	Address[31:0]			Data length[23:0]	Data
0x00	Bit 31=1	Module[6:0]	Register[24:0]	Length in words	Input array
1 byte	1 byte		3 bytes	3 bytes	Data length * 2 bytes

Example of Single Write

Write 0x33AA to the MAXIF (register 7) of the CM1K (module 1) 0x00 81 00 00 07 00 00 01 33 AA

Multiple Write

Write 4 consecutive byte components [11,12,13,14] of a neuron 0x00 81 00 00 01 00 00 02 11 12 13 14

Read command

Reserved	Address[31:0]			Data length[23:0]	Data
0x00	Bit 31=0	Module[6:0]	Register[24:0]	Length in words	Output array
1 byte	1 byte		3 bytes	3 bytes	Data length *2 bytes

Single Read

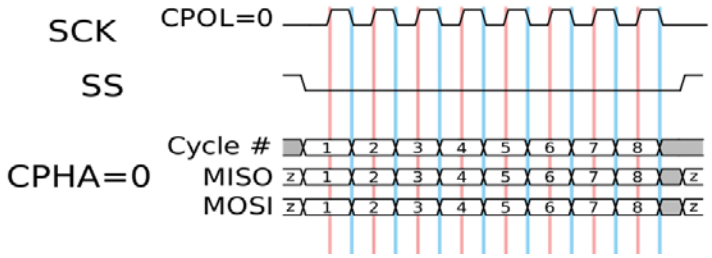
Read the MINIF register 6 of the CM1K (module 1) 0x00 01 00 00 06 00 00 01
Data is returned into 2 bytes or a word

Multiple Read

Read 8 consecutive byte components of a neuron 0x00 01 00 00 01 00 00 04
Data is returned into 8 bytes.

Timing Specifications

CPOL= 0 -> clock idle a logic level 0
CPHA= 0 -> Data sampled on rising edge of SCK
Recommended SCK frequency <= 10 MHz
General CSn Data pin #10 .
FPGA reconfiguration CSn= data pin #8



10 Updating the FPGA firmware

If the default firmware loaded in the Braincard is dated 08/01/2016 or later, the ability to reprogram the FPGA with a simple update of the flash memory is enabled. Otherwise, you will have to reprogram it with a JTAG programmer. Both methods are described below.

The version of the firmware is accessible by reading the register 0x0E and should return 16081 if it is the most recent firmware. Try the commands bcSPI.FPGArev() or bcSPI.read(0x0100000E).

Method 1: Flash memory update

At initialization of the BrainCard, the FPGA looks for a valid user configuration file in sector 8 of its Flash memory. If none is found, it loads the default configuration residing in sector 0.

Updating or upgrading the FPGA configuration is done very simply by copying a new configuration file (**bc_top.bin**) to an SD Card formatted as FAT32, running the FPGA_LoadConfig script supplied in the BrainCard Support Package and re-booting the Braincard.

Warning: If your BrainCard is mounted on an Intel Edison breakout board or a Raspberry PI board, use the SD card slot of these master boards. Otherwise, use the SD card slot of the BrainCard.

The Arduino examples contains two scripts to help you configure the FPGA of the BrainCard:
- FPGA_LoadConfig: Erase and load a new configuration file from an SD card to the sector 8 of the Flash memory. The file must be saved on the SD card as "bc_top.bin".
- FPGA_ResetConfig: Erase any user configuration file starting at sector 8 of the Flash memory

Method2: Reprogramming through JTAG

Loading a new configuration file on the Xilinx FPGA requires (1) a Xilinx JTAG programmer and (2) the iMPACT device configuration software from Xilinx.

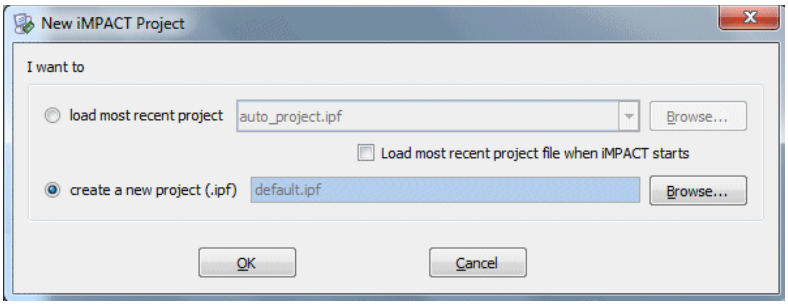
For the programmer we recommend the Xilinx HW-USB-II-G which comes with a flat cable allowing the programing of the BrainCard while plugged on an Arduino CPU. Other USB JTAG programmer are available but you need to verify compatibility with Xilinx programming software tools.

- Set the Jumper 8 to select the USB power supply (pin 1-2 connected)
- Connect the BrainCard to a USB power supply
- Plug the USB JTAG programmer into JTAG jumper and connect it to a second USB port

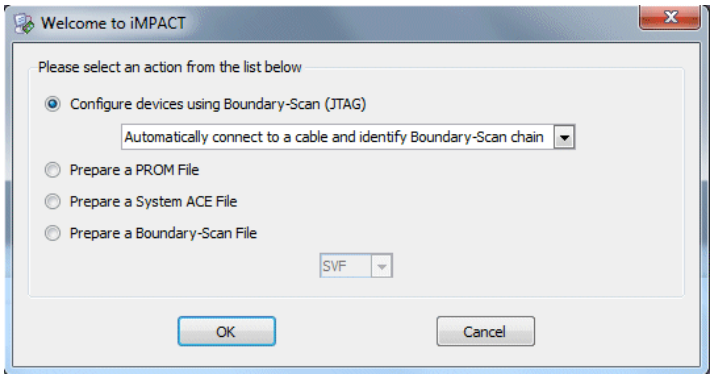
JTAG jumper			
1	NC	2	3.3 Volts
3	GND	4	TMS
5	GND	6	TCK
7	GND	8	TDO
9	GND	10	TDI
11	GND	12	NC
13	NC	14	NC

The iMPACT utility is supplied in the Xilinx ISE Design Tools and you have the choice to download the full product installation (6 GB), or just the Lab Tools (1 GB). The Xilinx Lab Tools is a standalone collection of the iMPACT device configuration and ChipScope Pro Analyzer tools. Standalone Lab Tools are intended for use in lab environments where the complete Xilinx ISE Design Suite toolset is not required.

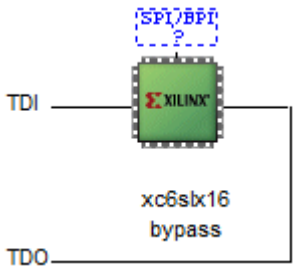
- Open the iMPACT software accessible through the Start Menu/Xilinx Design Tools/LabTools. Note that this might take a few seconds.
- Select the option to “Create a new project” and click OK.

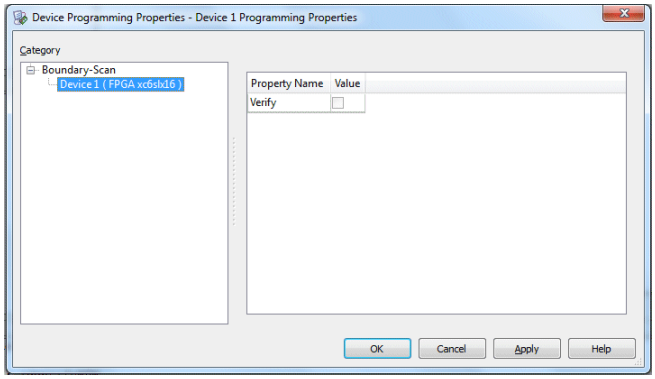


- Select the action to “Configure devices using Boundary-Scan” and click OK.

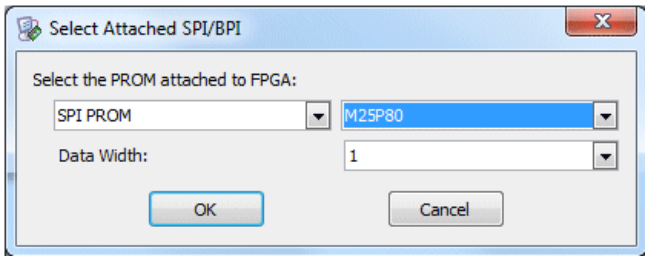


The Xilinx Spartan6 FPGA of the BrainCard should be identified and displayed as such on screen, but not the SPI Flash. Click OK in the Device Programming Properties panel.

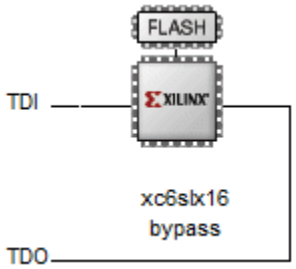




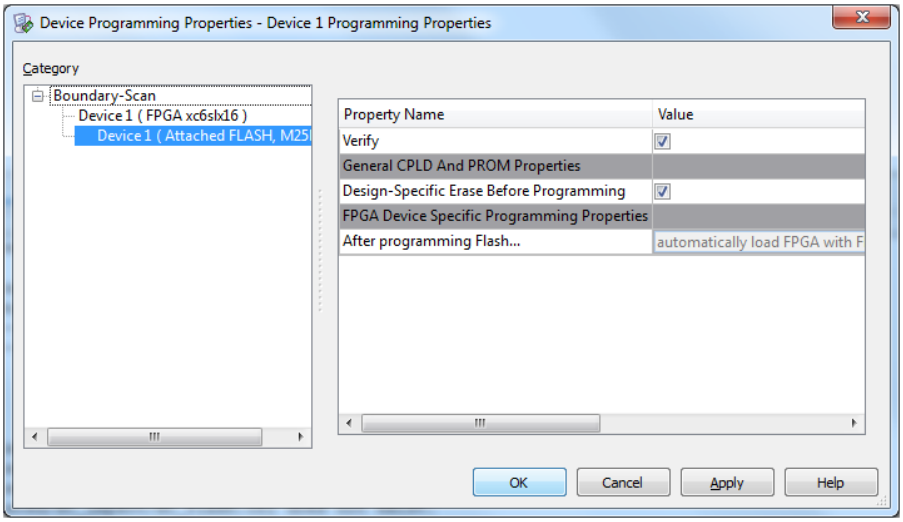
- Right click at the icon of the Xilinx and select the “Load a new configuration” command. In the next dialog box, select the file **bc_top.mcs** stored in the BrainCardBSP\FPGA_configuration folder.
- You will be then prompted to select the type of SPI PROM. The model on the BrainCard is M25P80 from Micron.



Upon proper configuration, the Flash icon will appear as identified.



- Right click at the Flash icon and select the “Program” command.
- The next panel identifies both the FPGA device and the Flash device. Click OK.



- Upon display of the message “Successful Programming”, you can disconnect the USB JTAG programmer.
- If applicable, disconnect the USB power supply and change the witch J8
- You are ready to interface the BrainCard with an Arduino, Raspberry PI or Intel Edison.

11 Troubleshooting SPI

The BrainCard does not power ON

- If the blue LED D1 is not lit, verify that the Jumper J18 is properly configured (between pin 2 and 3 for a USB power supply, pin 1 and 2 for Arduino power supply)
- Note that the LED ON is not sufficient to determine that the BrainCard gets enough power supply for stable operation. If experiencing instability in the behavior of the SPI communication, CM1K, and other components, try using external supply whenever possible over a cascaded power-over-USB coming from an Arduino board for example.

The BrainCard is not responding to SPI communications

- Verify that it is properly powered. The blue LED D1 should be lit
- Push the reset buttons of both the BrainCard and the processor card
- Run the Test_SPIComm and diagnose the type of the error (single Read/Write, multiple Read/Write)
- Test a different SPI speed (refer to your processor documentation)
- Test powering the platform with an external power supply

- On Arduino:
    - Verify that the SPI Chip Select pin 10 of the Arduino connector is not used by another shield
    - Verify that the pin 10 to 13 of the Arduino connector are reserved for the SPI communication of the BrainCard and not assigned anywhere in your code
    - The default SPI clock divider is one-quarter the frequency of the system clock (4 Mhz for the boards at 16 MHz). Depending on your Arduino processor, you might want to increase it using the function SetClockDivider.
- On Raspberry PI:
    - Do not forget to set SPI enabled under the configuration panel or via sudo raspi-config/advanced options
    - Speed should range from ~8kHz to 125MHz depending on the clock divider in use

If all fails, we can assume that the configuration file of the FPGA is corrupted and need reprogramming. Refer to the chapter “Updating the FPGA firmware” in this manual and reload the bc_top.bin or bc_top.mcs supplied in the Braincard BSP file.