Library User Manual

Damian Miralles

# Description

An open source implementation of a Code Division Multiple Access (CDMA) software correlator library that leverages Single Instruction Multiple Data (SIMD) is presented. We initially discuss the key aspects involved in the correlation operation for software radio applications. Afterward, we present the state of the art Application Programming Interface (API) that provides SIMD capable methods for each of the components in a correlation operation, including the first of its kind parallelized code and carrier generation using lookup tables and SIMD instructions. The library is developed using SIMD Intrinsic Instructions, which are a C type nomenclature offering access to the assembly instructions originally designed for the SIMD extensions in the processor. This design paradigm presents an advantage in terms of readability and simplified code development to accommodate future modifications. Recorded data was used with a standalone Global Navigation Satellite System (GNSS) software receiver where the methods hereby presented were tested and profiled to validate theoretical assumptions.

## Dependencies

**Raw signal file:** The GPS\_and\_GIOVE\_A-NN-fs16\_3676-if4\_1304.bin raw signal file can be found here:

1. <https://drive.google.com/file/d/1zrM0wjSQgH8hpJgw6_xE-E2yr8mNTEkn/view?usp=sharing>
2. <https://drive.google.com/open?id=1Jz07sTIdCOC557pjuQ5ODp1BuQ8h0-O8>

## Source Code

The source code available is organized as follows:

1. **src**: contains the SIMD library functions including multiplication, accumulation, and code and carrier generation with AVX2 and AVX512 support.
2. **doc**: contains the Doxygen style documentation of available functions in HTML format, with links to function parameters and return values if applicable. In order to generate the documentation user needs to run the make doc command.
3. **prof**: contains sample code on how to use the library functions in a simulated, standalone tracking scenario.
4. **plot**: contains binary files generated by prof source code to display tracking results values and validate new function additions or usage.
5. **build**: contains build directory for application using standard makefile components.
6. **data**: contains data dependencies needed to run the application
7. **install**: contains generated executables from prof source code and utility scripts to profile the code.

## Build code

For compilation the code uses traditional makefile syntaxes and nomenclature.

1 # Change directory to build

2 $ cd build/

3

4 # Compiles all executables available, this includes documentation

5 $ make

6

7 # clean generated executables

8 $ make clean

## Profile code

In Linux platforms, to enable profiling, obtain root privileges and run

1 # Obtain root priviledges and run

2 $root@root-pc:~# echo 0 > /proc/sys/kernel/yama/ptrace\_scope

This process is optional and allows user to verify working implementations of the code. Profiling has been automatized by running the install/profile\_cpu\_times.py script. Note that this implementation requires a working station with Intel VTune Amplifier installed on it. To profile, simply run:

1 # Change directory to install directory. Assumes executables present here

2 $ cd install/

3

4 # Configure script and run applications

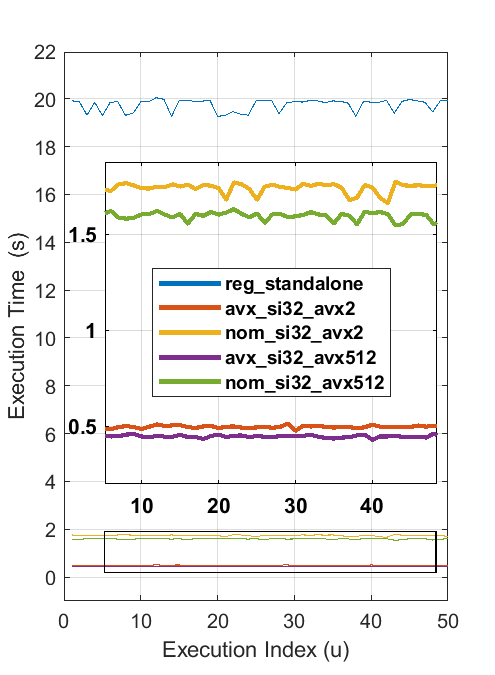
5 $ python profile\_cpu\_times.py

## Expected Results

### Profiling Results Analysis

To test the efficiency of the generated code, self-contained tracking executables were developed and profiled in an Intel(R) Core(TM) i9-7900X CPU @ 3.30GHz processor with support for AVX2 and AVX512. This platform also had installed the Intel Vtune Amplifier to serve as the profiling tools of the option. The testing/profiling procedure consisted of

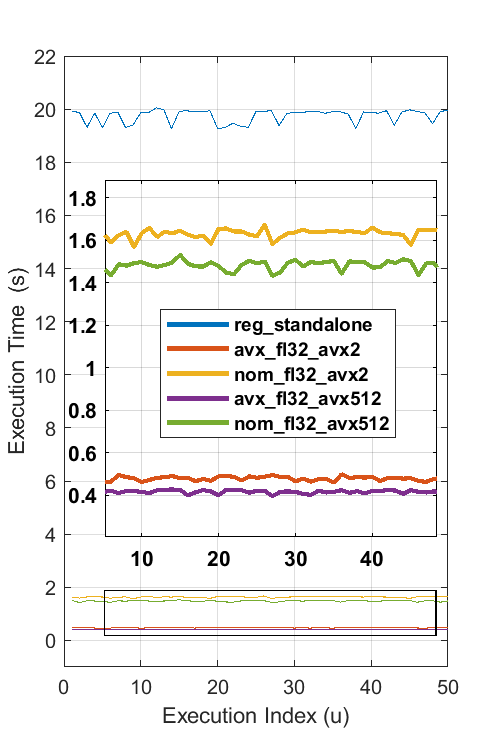
1. Running multiple executables that make calls to the API depending on its category, i.e., executables checking for AVX512 performance was only making function calls supporting the AVX512 function calls.
2. Dividing the test into two groups according to data type: (1) source files using single precision float for the correlation operation and (2) source files using 32-bits integers.
3. Dividing the test pool into three groups depending on the carrier and code generation: (1) regular code and carrier generation using C standard function calls, (2) nominal implementation with the DLUT method and (3) AVX implementation with the PLUT method for code and carrier generation. This last group also subdivides into AVX2 and AVX512 implementations.



**Fig. 1** Profiling results for si32 implementation in an Intel Skylake processor

Figure 1-2 display the execution times of running the executables fifty times in the targeted Skylake processor. All executables were compiled with the same optimization flags of -03, guarantying the maximum level of optimization possible while keeping a safe code generation. The executables in questions were labeled as

1. reg\_standalone: source code created using exclusively C function calls. This should mimic a generic implementation of a tracking algorithm and software correlator chip.
2. avx\_si32\_avx2: source code created using function calls that use AVX2 extensions and with code and carrier generation using the PLUT method with AVX2 support.
3. nom\_si32\_avx2: source code created using function calls that use AVX2 extensions for multiplication and accumulation but a nominal code and carrier generation using the DLUT method.
4. avx\_si32\_avx512: source code created using function calls that use AVX512 extensions and with code and carrier generation using the PLUT method and AVX512 support.
5. nom\_si32\_avx512: source code created using function calls that use AVX512 extensions for multiplication and accumulation but a nominal code and carrier generation using the DLUT method.



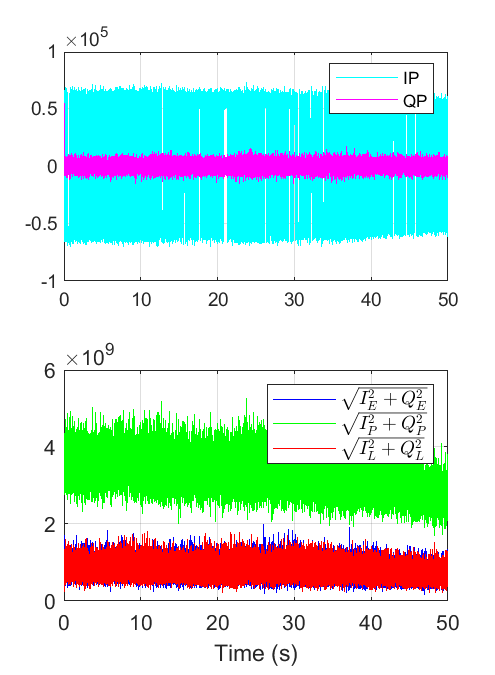
**Fig. 2** Profiling results for fl32 implementation in an Intel Skylake processor

**Table 1** Average execution time for one millisecond correlation

|  |  |
| --- | --- |
| Implementation | Average Time (s) |
| reg\_standalone | 6.581746666667e-05 |
| avx\_si32\_avx2 | 1.652866666667e-06 |
| nom\_si32\_avx2 | 5.806866666667e-06 |
| avx\_fl32\_avx2 | 1.598733333333e-06 |
| nom\_fl32\_avx2 | 5.440533333333e-06 |
| avx\_si32\_avx512 | 1.488400000000e-06 |
| nom\_si32\_avx512 | 5.321000000000e-06 |
| avx\_fl32\_avx512 | 1.391866666667e-06 |
| nom\_fl32\_avx512 | 4.943400000000e-06 |

### Validation of Tracking Results

Validation of all code created was done by adding the function call from the API to the self-contained tracking executables. Figure 3 shows the tracking results when running a highly efficient code that makes use of the AVX512 extensions for multiplication, addition, and carrier-code generation. The upper half of the figure displays the navigation data bits of a GPS L1 C/A signal ready to be decoded while the second half displays the correlation results of the operation.



**Fig. 3** Tracking results for PRN 22 using a real data set in a test scenario. Results include AVX512 carrier wave generation, AVX512 pseudorandom code generation, and parallelization of multiplication and addition operations. The top figure in the plot shows the decoded data bits in the navigation message while the bottom figure shows the correlation results.

# File Documentation

## avx2\_intrinsics.h File Reference

Provides C functions that implement Intel's AVX2 intrinsic.

#include "immintrin.h"

#include <math.h>

#include <stdio.h>

Include dependency graph for avx2\_intrinsics.h:

### Functions

* void **avx2\_nco\_si32** (int32\_t \*sig\_nco, const int32\_t \*lut, const int blk\_size, const double rem\_carr\_phase, const double carr\_freq, const double samp\_freq)

*Generates a NCO based on the Parallel Lookup Table (PLUT) method.*

* void **avx2\_nom\_nco\_si32** (int32\_t \*sig\_nco, const int32\_t \*lut, const int blk\_size, const double rem\_carr\_phase, const double carr\_freq, const double samp\_freq)

*Generates a nominal NCO based on the Direct Lookup Table (DLUT) method.*

* void **avx2\_nom\_code\_si32** (int32\_t \*ecode, int32\_t \*pcode, int32\_t \*lcode, const int32\_t \*cacode, const int blk\_size, const double rem\_code\_phase, const double code\_freq, const double samp\_freq)

*Generates an Early, late and Prompt code using SIMD Iinstructions.*

* void **avx2\_code\_si32** (int32\_t \*ecode, int32\_t \*pcode, int32\_t \*lcode, const int32\_t \*cacode, const int blk\_size, const float rem\_code\_phase, const float code\_freq, const float samp\_freq)

*Generates an Early, late and Prompt code using SIMD Iinstructions.*

* void **avx2\_nco\_fl32** (float \*sig\_nco, const float \*lut, const int blk\_size, const double rem\_carr\_phase, const double carr\_freq, const double samp\_freq)

*Generates a NCO based on the Direct Lookup Table (DLUT) method.*

* void **avx2\_nom\_nco\_fl32** (float \*sig\_nco, const float \*lut, const int blk\_size, const double rem\_carr\_phase, const double carr\_freq, const double samp\_freq)

*Generates a nominal NCO based on the direct lookup table approach.*

* void **avx2\_nom\_code\_fl32** (float \*ecode, float \*pcode, float \*lcode, const float \*cacode, const int blk\_size, const double rem\_code\_phase, const double code\_freq, const double samp\_freq)

*Generates an Early, Late and Propmt CA code based on the direct lookup table approach.*

* void **avx2\_code\_fl32** (float \*ecode, float \*pcode, float \*lcode, const float \*cacode, const int blk\_size, const float rem\_code\_phase, const float code\_freq, const float samp\_freq)

*Generates an Early, Late and Propmt CA code based on the Parallelized Lookup Table (PLUT) approach.*

### Detailed Description

Provides C functions that implement Intel's AVX2 intrinsic.

Code style and functionality based in the VOLK project.

##### Author:

Damian Miralles

Jake Johnson

##### Version:

4.1a

##### Date:

Jan 23, 2018

##### Precondition:

Make sure you have .bin files containing data and lookup tables

Definition in file **avx2\_intrinsics.h**.

### Function Documentation

#### void avx2\_code\_fl32 (float \* *ecode*, float \* *pcode*, float \* *lcode*, const float \* *cacode*, const int *blk\_size*, const float *rem\_code\_phase*, const float *code\_freq*, const float *samp\_freq*)

Generates an Early, Late and Propmt CA code based on the Parallelized Lookup Table (PLUT) approach.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *ecode* | Early CA code |
| out | *pcode* | Prompt CA code |
| out | *lcode* | Late CA code |
| in | *cacode* | Nominal satellite ranging code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_code\_phase* | Carrier phase remainder of the sinusoidal wave |
| in | *code\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **412** of file **avx2\_intrinsics.h**.

#### void avx2\_code\_si32 (int32\_t \* *ecode*, int32\_t \* *pcode*, int32\_t \* *lcode*, const int32\_t \* *cacode*, const int *blk\_size*, const float *rem\_code\_phase*, const float *code\_freq*, const float *samp\_freq*)

Generates an Early, late and Prompt code using SIMD Iinstructions.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *ecode* | Product vector storing the result of the multiplication |
| out | *pcode* | Product vector storing the result of the multiplication |
| out | *lcode* | Product vector storing the result of the multiplication |
| in | *cacode* | Source vector with factors to multiply |
| in | *blk\_size* | Source vector with factors to multiply |
| in | *rem\_code\_phase* | Number of points to Multiply in the operation |
| in | *code\_freq* | Number of points to Multiply in the operation |
| in | *samp\_freq* | Number of points to Multiply in the operation |

Definition at line **175** of file **avx2\_intrinsics.h**.

#### void avx2\_nco\_fl32 (float \* *sig\_nco*, const float \* *lut*, const int *blk\_size*, const double *rem\_carr\_phase*, const double *carr\_freq*, const double *samp\_freq*)

Generates a NCO based on the Direct Lookup Table (DLUT) method.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *sig\_nco* | Sinusoidal wave generated byt the NCO |
| in | *lut* | Lookup table to be use for the code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_carr\_phase* | Carrier phase remainder of the sinusoidal wave to be generated |
| in | *carr\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **263** of file **avx2\_intrinsics.h**.

#### void avx2\_nco\_si32 (int32\_t \* *sig\_nco*, const int32\_t \* *lut*, const int *blk\_size*, const double *rem\_carr\_phase*, const double *carr\_freq*, const double *samp\_freq*)

Generates a NCO based on the Parallel Lookup Table (PLUT) method.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *sig\_nco* | Sinusoidal wave generated byt the NCO |
| in | *lut* | Lookup table to be use for the code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_carr\_phase* | Carrier phase remainder of the sinusoidal wave to be generated |
| in | *carr\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **25** of file **avx2\_intrinsics.h**.

#### void avx2\_nom\_code\_fl32 (float \* *ecode*, float \* *pcode*, float \* *lcode*, const float \* *cacode*, const int *blk\_size*, const double *rem\_code\_phase*, const double *code\_freq*, const double *samp\_freq*)

Generates an Early, Late and Propmt CA code based on the direct lookup table approach.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *ecode* | Early CA code |
| out | *pcode* | Prompt CA code |
| out | *lcode* | Late CA code |
| in | *cacode* | Nominal satellite ranging code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_code\_phase* | Carrier phase remainder of the sinusoidal wave to be generated |
| in | *code\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **372** of file **avx2\_intrinsics.h**.

#### void avx2\_nom\_code\_si32 (int32\_t \* *ecode*, int32\_t \* *pcode*, int32\_t \* *lcode*, const int32\_t \* *cacode*, const int *blk\_size*, const double *rem\_code\_phase*, const double *code\_freq*, const double *samp\_freq*)

Generates an Early, late and Prompt code using SIMD Iinstructions.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *ecode* | Product vector storing the result of the multiplication |
| out | *pcode* | Product vector storing the result of the multiplication |
| out | *lcode* | Product vector storing the result of the multiplication |
| in | *cacode* | Source vector with factors to multiply |
| in | *blk\_size* | Source vector with factors to multiply |
| in | *rem\_code\_phase* | Number of points to Multiply in the operation |
| in | *code\_freq* | Number of points to Multiply in the operation |
| in | *samp\_freq* | Number of points to Multiply in the operation |

Definition at line **136** of file **avx2\_intrinsics.h**.

#### void avx2\_nom\_nco\_fl32 (float \* *sig\_nco*, const float \* *lut*, const int *blk\_size*, const double *rem\_carr\_phase*, const double *carr\_freq*, const double *samp\_freq*)

Generates a nominal NCO based on the direct lookup table approach.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *sig\_nco* | Sinusoidal wave generated byt the NCO |
| in | *lut* | Lookup table to be use for the code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_carr\_phase* | Carrier phase remainder of the sinusoidal wave to be generated |
| in | *carr\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **333** of file **avx2\_intrinsics.h**.

#### void avx2\_nom\_nco\_si32 (int32\_t \* *sig\_nco*, const int32\_t \* *lut*, const int *blk\_size*, const double *rem\_carr\_phase*, const double *carr\_freq*, const double *samp\_freq*)

Generates a nominal NCO based on the Direct Lookup Table (DLUT) method.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *sig\_nco* | Sinusoidal wave generated byt the NCO |
| in | *lut* | Lookup table to be use for the code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_carr\_phase* | Carrier phase remainder of the sinusoidal wave to be generated |
| in | *carr\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **96** of file **avx2\_intrinsics.h**.

## avx512\_intrinsics.h File Reference

Provides C functions that implement Intel's AVX512 intrinsic.

#include "immintrin.h"

#include <math.h>

#include <stdio.h>

Include dependency graph for avx512\_intrinsics.h:

### Functions

* void **avx512\_nco\_si32** (int \*sig\_nco, const int \*lut, const int blk\_size, const double rem\_carr\_phase, const double carr\_freq, const double samp\_freq)

*Generates a NCO based on the Paralelized Lookup Table (PLUT) approach.*

* void **avx512\_nom\_nco\_si32** (int \*sig\_nco, const int \*lut, const int blk\_size, const double rem\_carr\_phase, const double carr\_freq, const double samp\_freq)

*Generates a nominal NCO based on the Direct Lookup Table (DLUT) approach.*

* void **avx512\_nom\_code\_si32** (int \*ecode, int \*pcode, int \*lcode, const int \*cacode, const int blk\_size, const double rem\_code\_phase, const double code\_freq, const double samp\_freq)

*Generates an Early, Late and Propmt CA code based on the direct lookup table approach.*

* void **avx512\_code\_si32** (int \*ecode, int \*pcode, int \*lcode, const int \*cacode, const int blk\_size, const float rem\_code\_phase, const float code\_freq, const float samp\_freq)

*Generates an Early, Late and Propmt CA code based on the Parallelized Lookup Table (PLUT) approach.*

* void **avx512\_nco\_fl32** (float \*sig\_nco, const float \*lut, const int blk\_size, const double rem\_carr\_phase, const double carr\_freq, const double samp\_freq)

*Generates a NCO based on the direct lookup table approach with SIMD intrinsic and generates the carrier as fl32 type.*

* void **avx512\_nom\_nco\_fl32** (float \*sig\_nco, const float \*lut, const int blk\_size, const double rem\_carr\_phase, const double carr\_freq, const double samp\_freq)

*Generates a nominal NCO based on the direct lookup table approach as fl32 type.*

* void **avx512\_nom\_code\_fl32** (float \*ecode, float \*pcode, float \*lcode, const float \*cacode, const int blk\_size, const double rem\_code\_phase, const double code\_freq, const double samp\_freq)

*Generates an Early, Late and Propmt CA code based on the direct lookup table approach.*

* void **avx512\_code\_fl32** (float \*ecode, float \*pcode, float \*lcode, const float \*cacode, const int blk\_size, const float rem\_code\_phase, const float code\_freq, const float samp\_freq)

*Generates an Early, Late and Propmt CA code based on the direct lookup table approach with SIMD instructions.*

### Detailed Description

Provides C functions that implement Intel's AVX512 intrinsic.

Code style and functionality based in the VOLK project.

##### Author:

Damian Miralles

Jake Johnson

##### Version:

4.1a

##### Date:

Jan 23, 2018

##### Precondition:

Make sure you have .bin files containing data and lookup tables.

##### Note:

Functions in the file must target AVX512 enabled platforms.

Definition in file **avx512\_intrinsics.h**.

### Function Documentation

#### void avx512\_code\_fl32 (float \* *ecode*, float \* *pcode*, float \* *lcode*, const float \* *cacode*, const int *blk\_size*, const float *rem\_code\_phase*, const float *code\_freq*, const float *samp\_freq*)

Generates an Early, Late and Propmt CA code based on the direct lookup table approach with SIMD instructions.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *ecode* | Early CA code |
| out | *pcode* | Prompt CA code |
| out | *lcode* | Late CA code |
| in | *cacode* | Nominal satellite ranging code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_code\_phase* | Carrier phase remainder of the sinusoidal wave |
| in | *code\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **540** of file **avx512\_intrinsics.h**.

#### void avx512\_code\_si32 (int \* *ecode*, int \* *pcode*, int \* *lcode*, const int \* *cacode*, const int *blk\_size*, const float *rem\_code\_phase*, const float *code\_freq*, const float *samp\_freq*)

Generates an Early, Late and Propmt CA code based on the Parallelized Lookup Table (PLUT) approach.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *ecode* | Early CA code |
| out | *pcode* | Prompt CA code |
| out | *lcode* | Late CA code |
| in | *cacode* | Nominal satellite ranging code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_code\_phase* | Carrier phase remainder of the sinusoidal wave |
| in | *code\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **181** of file **avx512\_intrinsics.h**.

#### void avx512\_nco\_fl32 (float \* *sig\_nco*, const float \* *lut*, const int *blk\_size*, const double *rem\_carr\_phase*, const double *carr\_freq*, const double *samp\_freq*)

Generates a NCO based on the direct lookup table approach with SIMD intrinsic and generates the carrier as fl32 type.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *sig\_nco* | Sinusoidal wave generated byt the NCO |
| in | *lut* | Lookup table to be use for the code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_carr\_phase* | Carrier phase remainder of the sinusoidal wave |
| in | *carr\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **384** of file **avx512\_intrinsics.h**.

#### void avx512\_nco\_si32 (int \* *sig\_nco*, const int \* *lut*, const int *blk\_size*, const double *rem\_carr\_phase*, const double *carr\_freq*, const double *samp\_freq*)

Generates a NCO based on the Paralelized Lookup Table (PLUT) approach.

Implementation uses the SIMD AVX512 intrinsics.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *sig\_nco* | Sinusoidal wave generated byt the NCO |
| in | *lut* | Lookup table to be use for the code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_carr\_phase* | Carrier phase remainder of the sinusoidal wave |
| in | *carr\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **27** of file **avx512\_intrinsics.h**.

#### void avx512\_nom\_code\_fl32 (float \* *ecode*, float \* *pcode*, float \* *lcode*, const float \* *cacode*, const int *blk\_size*, const double *rem\_code\_phase*, const double *code\_freq*, const double *samp\_freq*)

Generates an Early, Late and Propmt CA code based on the direct lookup table approach.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *ecode* | Early CA code |
| out | *pcode* | Prompt CA code |
| out | *lcode* | Late CA code |
| in | *cacode* | Nominal satellite ranging code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_code\_phase* | Carrier phase remainder of the sinusoidal wave |
| in | *code\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **500** of file **avx512\_intrinsics.h**.

#### void avx512\_nom\_code\_si32 (int \* *ecode*, int \* *pcode*, int \* *lcode*, const int \* *cacode*, const int *blk\_size*, const double *rem\_code\_phase*, const double *code\_freq*, const double *samp\_freq*)

Generates an Early, Late and Propmt CA code based on the direct lookup table approach.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *ecode* | Early CA code |
| out | *pcode* | Prompt CA code |
| out | *lcode* | Late CA code |
| in | *cacode* | Nominal satellite ranging code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_code\_phase* | Carrier phase remainder of the sinusoidal wave |
| in | *code\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **142** of file **avx512\_intrinsics.h**.

#### void avx512\_nom\_nco\_fl32 (float \* *sig\_nco*, const float \* *lut*, const int *blk\_size*, const double *rem\_carr\_phase*, const double *carr\_freq*, const double *samp\_freq*)

Generates a nominal NCO based on the direct lookup table approach as fl32 type.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *sig\_nco* | Sinusoidal wave generated byt the NCO |
| in | *lut* | Lookup table to be use for the code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_carr\_phase* | Carrier phase remainder of the sinusoidal wave |
| in | *carr\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **459** of file **avx512\_intrinsics.h**.

#### void avx512\_nom\_nco\_si32 (int \* *sig\_nco*, const int \* *lut*, const int *blk\_size*, const double *rem\_carr\_phase*, const double *carr\_freq*, const double *samp\_freq*)

Generates a nominal NCO based on the Direct Lookup Table (DLUT) approach.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *sig\_nco* | Sinusoidal wave generated byt the NCO |
| in | *lut* | Lookup table to be use for the code |
| in | *blk\_size* | Total number of elements in the sig\_nco vector |
| in | *rem\_carr\_phase* | Carrier phase remainder of the sinusoidal wave |
| in | *carr\_freq* | Carrier frequency of the sinusoidal wave |
| in | *samp\_freq* | Sampling frequency of the signal to be generated |

Definition at line **101** of file **avx512\_intrinsics.h**.

## mmx\_intrinsics.h File Reference

Provides C functions that implement Intel's MMX intrinsic.

#include "mmintrin.h"

#include <stdio.h>

Include dependency graph for mmx\_intrinsics.h:

### Detailed Description

Provides C functions that implement Intel's MMX intrinsic.

Based off VOLK kernel functions.

##### Author:

Jake Johnson

##### Version:

4.1a

##### Date:

Jan 23, 2018

##### Precondition:

Make sure you have .bin files containing data and lookup tables

Definition in file **mmx\_intrinsics.h**.

## read\_bin.h File Reference

Utility functions for reading data from data files.

#include <stdio.h>

#include <stdlib.h>

Include dependency graph for read\_bin.h:

### Functions

* double **getDoubleFromFile** (char file\_name[1000])
* int **getIntFromFile** (char file\_name[1000])
* double **getcaCodeFromFile** (char file\_name[1000], double \*output\_array)
* void **getcaCodeFromFileAsInt** (char file\_name[1000], int \*output\_array)
* void **getcaCodeFromFileAsFloat** (char file\_name[1000], float \*output\_array)
* void **getDataFromFile** (char file\_name[1000], double \*output\_array, int size)

### Detailed Description

Utility functions for reading data from data files.

##### Author:

Damian Miralles

Jake Johnson

##### Version:

4.1a

##### Date:

Jan 23, 2018

Definition in file **read\_bin.h**.

### Function Documentation

#### double getcaCodeFromFile (char *file\_name*[1000], double \* *output\_array*)

Definition at line **53** of file **read\_bin.h**.

#### void getcaCodeFromFileAsFloat (char *file\_name*[1000], float \* *output\_array*)

Definition at line **97** of file **read\_bin.h**.

#### void getcaCodeFromFileAsInt (char *file\_name*[1000], int \* *output\_array*)

Definition at line **75** of file **read\_bin.h**.

#### void getDataFromFile (char *file\_name*[1000], double \* *output\_array*, int *size*)

Definition at line **119** of file **read\_bin.h**.

#### double getDoubleFromFile (char *file\_name*[1000])

Definition at line **13** of file **read\_bin.h**.

#### int getIntFromFile (char *file\_name*[1000])

Definition at line **33** of file **read\_bin.h**.