

ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Dottorato di Ricerca in

MONITORAGGIO E GESTIONE DELLE STRUTTURE E DELL'AMBIENTE - SEHM2

XXXIII Ciclo

Settore Concorsuale: 08/B1 - Geotecnica

Settore Scientifico Disciplinare: ICAR/07 - Geotecnica

A GENERAL FRAMEWORK FOR IOT-READY GEOTECHNICAL OBJECTS

Presentata da: Giacomo Mazzoni

Coordinatore di Dottorato

Prof. Ing. Alessandro Marzani

Supervisore

Prof. Ing. Laura Tonni

Co-supervisore

Dott. Giulio Dal Forno

Esame finale anno 2021

Foreword

In this thesis the outcomes of a 3-year research and development project hosted at ENVIA in collaboration with SEHM2 (Structural and Environmental Health Monitoring & Management) PhD program at Alma Mater Studiorum - University of Bologna are presented.

The project focuses on the relationship between current IoT trends and geotechnics within the civil engineering context. A simple framework for geotechnical data exchange tailored to IoT applications, potentially suitable for both small to large scale structural and infrastructural scenarios, has been prototyped. Within such scenarios further a special case of data generation from sensor-enabled geosynthetics has been explored.

The framework is primarily based on the data exchange open standards of Geographical Information Systems (GIS) and Building Information Modeling (BIM). In particular, it entails an extension of the current AGS4 data exchange format through a novel AGS-JSON schema that is specifically tailored to store and manage data streams from sensors.

Additionally, different modelling and analytics strategies, based both on FEA procedures and Machine Learning algorithms, have been developed to provide the necessary data-to-knowledge tools to complement the framework. In one specific case (the smart foundation prototype) a contribution to the development of a SaaS, which is currently in its beta-testing phase and available to a selected number of engineers in the Australasian region, has been provided.

Finally, the most prominent “environmental” factors that may challenge the opportunity of ubiquitous and heterogeneous geotechnical data sources, including sensor-enabled geotechnical objects, have been investigated. In particular, the use of blockchain technology to provide an additional security layer based on the “data transactions” logic has been investigated. Furthermore, general economical considerations based on the outcomes of this research have been reported, emphasizing the potential benefits of sensor-enabled geotechnical objects within the fast-growing IoT panorama.

Table of contents

Foreword.....	1
Table of contents.....	3
List of acronyms.....	5
List of symbols.....	6
1. Internet of things and geotechnics.....	7
1.1 General trends of the construction industry in a data-driven economy.....	8
1.2 Data mining in the AEC industry.....	11
1.3 The idle capacity of geological and geotechnical data.....	12
1.4 Geotechnics and the Geographical Information Systems.....	15
1.5 Geotechnics and the Building Information Modelling.....	16
1.6 The potential role of sensor-enabled geotechnical objects.....	17
2. Data sources.....	19
2.1 Typical geotechnical data sources.....	19
2.2 “Smart” objects vs. regular geotechnical monitoring.....	22
2.3 Sensor-enabled geosynthetics.....	23
2.4 Smart geocell prototype: physical components.....	26
2.4.1 Sensors.....	26
2.4.2 Wiring, enclosure and installation.....	28
2.4.3 Powering and networking.....	28
3. Data management.....	30
3.1 The pitfall of dispersive data.....	30
3.2 The GIS approach.....	31
3.3 The BIM approach.....	34
3.4 Smart geocell prototype: edge computing.....	35
3.4.1 Data normalization and standardization.....	35
3.4.2 Low-level thresholds handling.....	40
4. Modelling and analytics.....	42
4.1 Model-based vs. Data-driven approach.....	42
4.2 Distributed numerical modelling: the smart foundation.....	43

4.3	Distributed Machine Learning.....	49
4.4	Smart geocell prototype: machine learning.....	50
4.4.1	Status classifier	50
4.4.2	Predictive maintenance regressor.....	52
5.	Data security.....	54
5.1	Centralized vs. Decentralized data.....	54
5.2	Cryptography and Blockchains.....	55
5.3	Policies and regulations	57
5.4	Data ethics.....	60
6.	Economical considerations.....	61
6.1	The general paradigm of data value.....	61
6.2	The costs of geotechnical data mining	62
6.3	The benefits of geotechnical data mining	65
7.	Concluding remarks.....	68
7.1	Review of the research.....	68
7.2	Discussion.....	70
7.3	Future research and developments.....	72
	Acknowledgements	74
	References.....	75

LIST OF ANNEXES

Annex 1 - Hardware description

Annex 2 - Software design principles and strategies

List of acronyms

AEC	Architecture Engineering Construction
AGS	Association of Geotechnical and Geoenvironmental Specialists
API	Application Programming Interface
BIM	Building Information Modelling
CPU	Central Processing Unit
DIGGS	Data Interchange for Geotechnical and Geoenvironmental Specialists
DT	Digital Twin, Decision Tree
EDF	Électricité de France
EU	European Union
FEA	Finite Element Analysis
GIS	Geographical Information System
GPS	Global Positioning System
HM	High Modulus
IFC	Industry Foundation Classes
IoT	Internet of Things
IT	Information Technology
JSON	Javascript Object Notation
MEMS	Micro Electromechanical System
ML	Machine Learning
OT	Operation Technology
PCA	Principal Component Analysis
PSD	Power Spectral Density
RMS	Root Mean Square
RF	Random Forest
SaaS	Software as a Service
SHM	Structural Health Monitoring

Note: Acronyms are presented in alphabetical order

List of symbols

a	Linear acceleration
ϵ	Correction (small) factor
P	Modified periodogram from discrete Fourier transform
K	Number of batches
A	Target matrix (for PCA)
U	Matrix with orthonormal columns of dimension $m \times k$
V	Matrix with orthonormal columns of dimensions $n \times k$
Σ	Diagonal matrix
n	Number of samples
k	Number of features
x	Sample feature value, Predictor value, Generic variable
μ	Cluster centroid feature value
C	Cluster
Q	Data subset (batch)
G	Impurity
H	Impurity base function
R	Observation region
N	Observation number
p	Probability
X	Training data
L	Loss function (linear regressor)
y	Response variable
α	Calibration parameter
β	Calibration parameter

Note: Symbols are presented in order of appearance

1. Internet of things and geotechnics

The Internet of Things (IoT) is a broad and sometimes controversial technological trend, which is becoming transversal to numerous aspects of the modern global economy. Such trend is necessarily galloping at global scale as IoT is intrinsically related to the current level of interconnection, which is the highest humanity has ever experienced.

Pushed by the ubiquity of IoT, many industries, including low-tech ones, are experiencing a surge in technical and regulatory instruments aiming to its adoption. However, while there is consensus among a variety of stakeholders that IoT would bring outstanding social and economical results, there is still uncertainty in the very definition of the term as well as in the definition of the underlying philosophy.

Some authors suggest a general and positive definition for IoT based on the desirable convergence between *people, things, processes* and *data* towards a data-driven economy (Rayes, 2019). Others outline the duality between network-enabled capabilities and network-induced constraints in a super connected physical world (Brous, 2020; Zuboff, 2018).

With this second, even-tempered definition in mind, one might argue that a specific industry would endorse the IoT logic and instruments due to their expected end-state benefits, while underestimating the structural changes and the impacts this adoption is going to produce.

The adoption course in the case of the AEC industry is following the rise of Building Information Modeling (BIM) and its progressive merging with Geographical Information Systems (GIS). BIM first adoption efforts at scale were promoted in the UK in the early 2000s with an initial focus on architectural and structural design. It soon became clear that BIM could have represented a leap forward for the whole

industrial process, as AEC suffered from chronic inefficiencies due to a lack of proper communication when compared to other, more *integrated* industries.

This introductory chapter outlines some general features and trends of the IoT in the data-driven economy. It focuses briefly on the AEC industry's most notable data management systems (BIM and GIS as anticipated) and, finally, it presents some broad reasoning about the current and potential role of data, especially geotechnical data, in the construction sector information value chain.

1.1 General trends of the construction industry in a data-driven economy

As many authors point out (Rayes, 2019), the general trend of the *Fourth Industrial Revolution*, which is dominated by the IoT paradigm, is propelled by the convergence of Operation Technology (OT) and Information Technology (IT). This convergence brings real-time data from operations closer to the expertise and technologies capable of using such data to make decisions and automate feedback instructions. Thanks to a dramatic increase in data quantity and granularity (Bramer, 2016), information itself has become a raw material for production in many industrial sectors. A schema for the general process of data-to-knowledge exploiting data as raw material is shown in Figure 1.

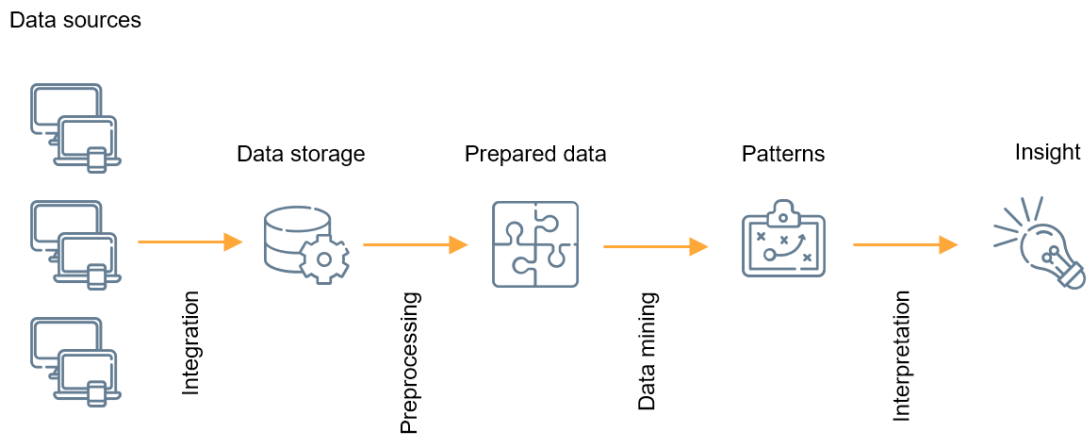


Figure 1 - General schema for the data-to-knowledge process

As anticipated, the process applies to a wide range of industries with some notable examples of disruptive achievements. In the AEC industry this logic is most prominent in the *smart building* philosophy, in which sensor-enabled built items are deemed to optimize a plethora of routine and maintenance operations throughout the life of a structure or an infrastructure. For both buildings and infrastructures this might include, for instance:

- Safety monitoring and alerting (e.g. fire, noise, vibration, flooding, air pollution, earthquake early warning, etc.)
- Smart energy management (incl. lighting, natural gas, renewable energy production, etc.)
- Predictive maintenance

As for data integration (see Fig. 1), the AEC industry relies on two major means of data standardization:

- The Geographical Information System standards (GIS)
- The Building Information Modeling standards (BIM)

The latter being the youngest and more complex database format built around the Industry Foundation Class format (ISO 16739, acronym IFC) and currently in the way of converging with GIS.

Nowadays BIM is undergoing a rapid evolution that is shifting the underlying logic from project-based collaboration to data integration (Ademci, 2018). This is usually identified as a shift from BIM to integrated BIM or iBIM (see Fig. 2).

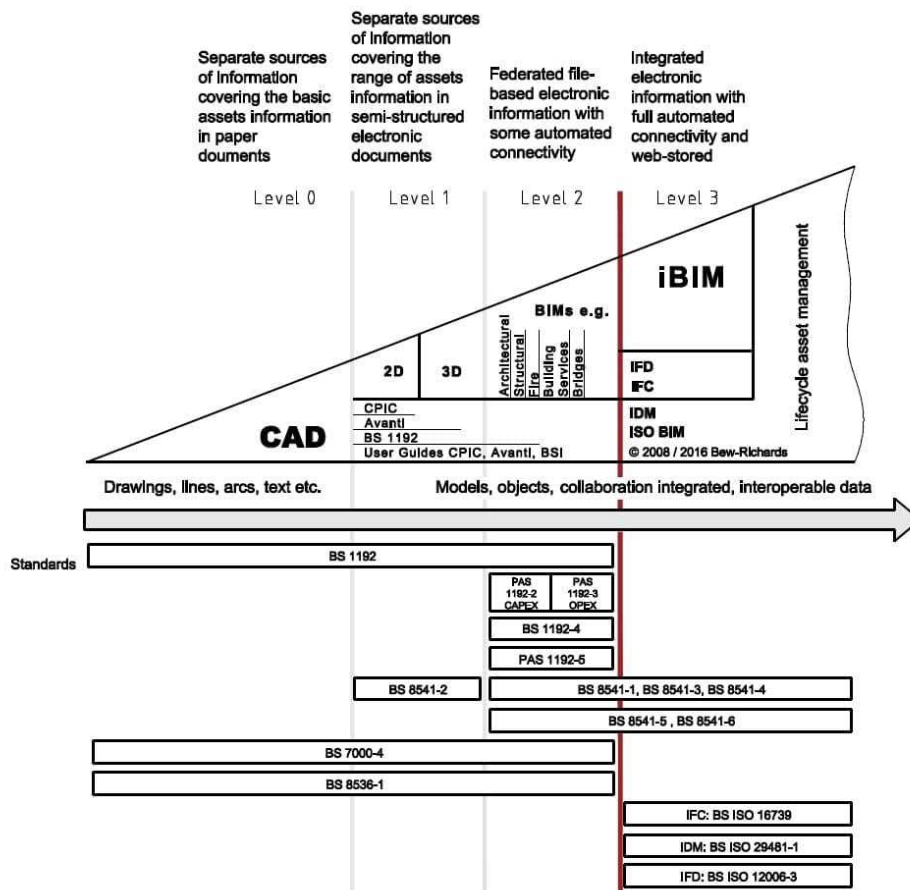


Figure 2 - BIM evolution chart

As broadly stated in the recently delivered EN ISO 19650:2019, BIM is *about getting benefit through better specification and delivery of just the right amount of information concerning the design, construction and management of buildings and infrastructure, using appropriate technology tools*. Hence it is quite clear that BIM's prime purpose is to make clarity in the increasing data complexity of the AEC industry and promote value extraction from it.

IoT lives in the realm of Level 3 BIM (iBIM), where a fully functional and widely adopted database standard is capable of handling a considerable amount of diverse data sources to convey any sort of valuable information concerning a building or an infrastructural asset.

1.2 Data mining in the AEC industry

As anticipated in section 1.1, data are important raw materials in the Fourth Industrial Revolution paradigm. The term *data mining* is then an appropriate way to describe such attitude towards maximizing data collection and data quality to extract the highest value from this resource.

Unlike the most prominent sectors in this latest industrial revolution, the AEC industry does not rely massively on *big data* to enhance business performances yet (Ahmed, 2018). This might be due to several hindering factors, including:

- Low adoption rates for BIM, GIS and data standardization protocols in general
- Low level of global construction standard harmonization
- Overall low technological level of the industry

Although no clear path for data mining in the AEC sector has been set so far, it is understood that the industry is now sufficiently mature to benefit from an increasing attention to information quality and management (EU BIM Taskgroup, 2017). It is the writer own opinion that this includes his specific fields of interest, namely applied geology and geotechnical engineering.

1.3 The idle capacity of geological and geotechnical data

Geological and geotechnical data are usually collected at the beginning of every civil engineering project. Those data are of the utmost importance in the early phases of a project to allow for strategic decision regarding e.g. foundations and ground improvement activities. They are also important to evaluate and mitigate serious risks related to geo-hazards (earthquakes, landslides, expansive soils, etc.). The lack of sound geological and geotechnical data is often regarded as a major cause of unexpected costs, damages and fatalities; plus it prevents professionals from adopting risk mitigation strategies based on robust statistics (Christian, 2011).

The most interesting aspect of this chronic lack of *usable* data about the underground is that, at least in most cases, the information (including direct sounding, laboratory testing, geophysical testing, etc.) is present but totally *unusable* or only *partially usable* by the engineers. This is especially the case for historical data.

Moreover, in a number of different scenarios including high-risk contexts, geological and geotechnical monitoring is simply neglected, and permanent sensor-enabled geotechnical elements (e.g. sensor-enabled piles) are seldomly part of routine monitoring protocols.

If tunneling works are not accounted for, in the writer's professional experience geological and geotechnical monitoring is more often set up *a posteriori* when some

sort of damage has already occurred both during construction or after construction completion. And even when monitoring takes place, this is usually relegated to a *data niche* that lacks in standardization protocols and ultimately provides very little value for the investors as well as for the contractors.

Standardization is surely one relevant issue, even though some interesting standards for geological and geotechnical data have reached maturity and others are gaining momentum at present (see Fig. 3). To cite a few of the most relevant data exchange formats:

- GeoSciML from the Open Geospatial Consortium (OGC)
- AGS data format from the (UK) Association of Geotechnical & Geo-environmental Specialists
- DIGGS data format from ASCE Geo Institute (USA)

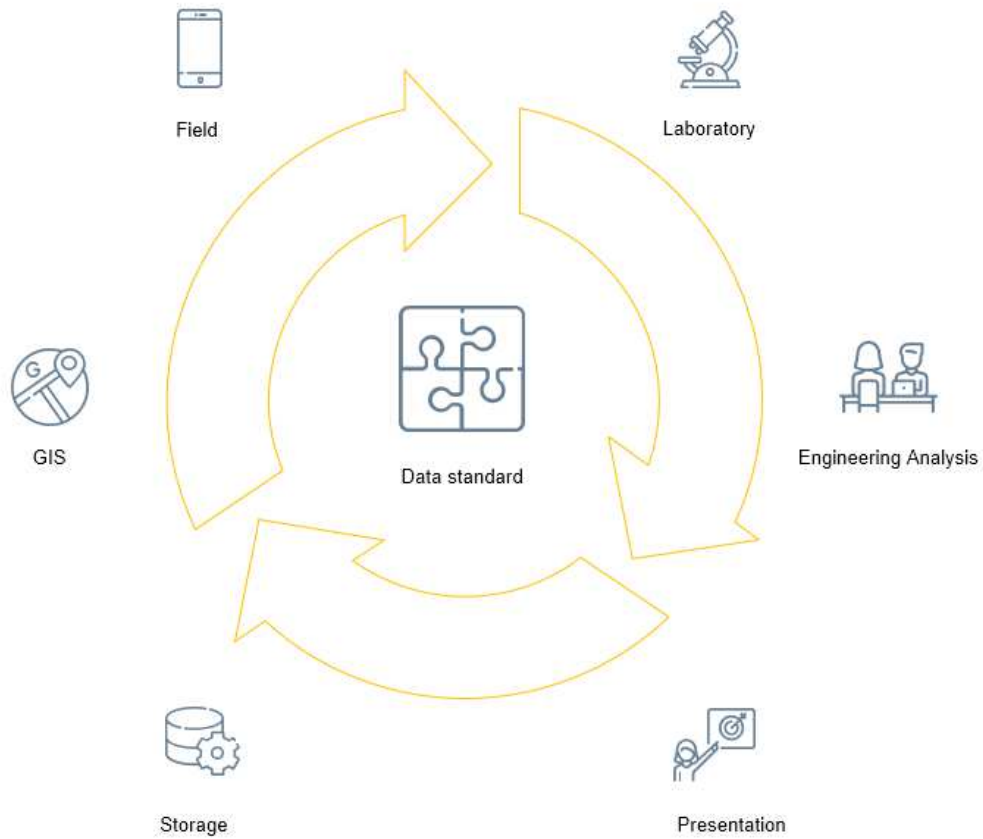


Figure 3 - Benefits of standardization in geotechnical data management

The idle capacity of geological and geotechnical data is then somehow universally recognized, even though it still gets little attention from the industry and its processes.

Geographical Information Systems (GIS) are frequently and commercially used to manage geological and geotechnical factual and interpreted data, even though, at least in most cases, open standards are omitted in favor of proprietary management data formats.

An important paradigm change in terms of interoperability could be represented by the current attempts within Building Smart International to introduce strong subsurface data standardization within the Building Information Modeling developing framework.

1.4 Geotechnics and the Geographical Information Systems

As far as the relationship between geotechnics and GIS is concerned, the market offers a relatively wide variety of software capable of handling geological and geotechnical information in a geospatial fashion. Usually two main categories of management procedures are found:

- GIS for structural geology
- GIS for geotechnical *in situ* and laboratory testing and geotechnical monitoring management

The first procedure is most tailored to producing visualization aids based on direct sounding and geophysical testing and focuses mainly on structural geology. This includes mapping, 3D spatial interpolation and uncertainty management. This kind of software is usually very well suited for large scale project such as tunnels, offshore installations, etc.

On the other hand, another class of well-established software is conceived to geotechnical *in situ* and laboratory testing management, resulting in wider use cases, ranging from small to large scale civil engineering projects. This second data management schema is becoming more and more integrated into complex automated or semi-supervised workflows, starting from field data acquisition to geotechnical model generation to geotechnical numerical simulations.

In both of the aforementioned logics, the underlying feature of GIS that is so well suited for managing information about the underground, is the ability to offer spatial correlation tools that enormously facilitate data manipulation and allow valuable information extraction from bulky datasets (potentially including historical data).

1.5 Geotechnics and the Building Information Modelling

As anticipated in section 1.1 and as shown in some recent pioneering research (Fabozzi S., 2020; Huang M. Q., 2020), the Building Information Modeling (BIM) is an extremely attractive opportunity to centralize and extract value from data produced at the project scale. As far as geotechnics is concerned, this means establishing some adequately broad and complete criteria to store and share information about soils/rocks, underground fluids and, especially, underground structures and infrastructures.

According to a recent survey (Tawelian, 2016), it appears that the current status of *geotechnical BIM* is still quite immature, however, most of the interviewees at that time agreed that, as applied geology and geotechnical professionals, they would benefit from the adoption of the communication standards of BIM.

	Helpful	Harmful
Internal (organization)	Strengths: <ul style="list-style-type: none"> • BIM process familiar to most of engineers • Familiarity with the current geotechnical data standards • Potential for decrease in time and costs for projects 	Weaknesses: <ul style="list-style-type: none"> • Current BIM does not include a strong geotechnical data exchange protocol (yet) • Current applications do not incorporate most of geotechnical design elements • Small companies may struggle with initial investment in training and software/hardware
External (environment)	Opportunities: <ul style="list-style-type: none"> • Training based on successful application of the process to relevant projects • Relatively small number of geotechnical data types • Reducing risks 	Threats: <ul style="list-style-type: none"> • Segmentation of geotechnical BIM protocols • Collaborative working may be slowed down • Increase in project costs

Table 1 - SWOT analysis for BIM applied to geotechnics

Although none of the aforementioned BIM-compliant storage and sharing criteria are present in the previous and current versions of the IFC standard (current version IFC4.1), attempts are being made to include some new geological and geotechnical distinctive features in the IFC5 development standard. Those attempts are aiming to integrate some existing and industry-wide data exchange formats (see section 1.3) within the BIM framework as well as to define new standards for e.g. interpreted data (including risk assessment) and monitoring. This work, carried out mainly within the Infrastructure Room in the Building Smart International panel, is also expected to deliver standards and tools for BIM and GIS integration.

One apparently minor and yet lingering desirable aspect of the developing geotechnical BIM emerges in the same survey, especially when the authors summarize their findings in a SWOT chart (see Table 1). This aspect regards what the authors (Tawelian, 2016) address to as *buried services and underground structures*, i.e. elements relevant to the project that lay underground and interact/interfere with geotechnical works.

The idea of extracting valuable information from components laying underground stands at the basis of this work. The goal, as engineering services providers, would be to manage and manipulate data from sensor-enabled *geotechnical objects*, to take advantage of the normalization ensured by the BIM framework and, ultimately, draw value from otherwise idle information sources.

1.6 The potential role of sensor-enabled geotechnical objects

Although Health Monitoring is not a new topic for some distinctive, geotechnical-intensive civil engineering activities (incl. deep excavations, roadways and airfields), the vast majority of all geotechnical remote sensing installation focuses on project

variable and constrains rather than pure data extraction. It is quite common to have optical displacement monitoring during deep excavation works, but it is not so common to have inertial monitoring systems systematically embedded in D-wall reinforcements. It is even less common to have integrated monitoring systems that can produce valuable information for all the involved stakeholders (designers, contractors, HS, developers, etc.).

It is the writer's opinion that the potential role of sensors enabled geotechnical objects, such as precast and cast-in-place deep foundations, ground anchors, geosynthetics, tunnel liners, etc. is currently underestimated and relatively simple to achieve, at least in medium-to-large structural and infrastructural scenarios.

Especially geosynthetics, which are a product of a relatively highly specialized and precision industry, seem the most suitable first candidate for a transition towards natively smart geotechnical objects. This research has explored the potential of one class of geosynthetics, namely geocells, to be equipped with low-cost, consumer grade MEMS sensors and to be able to *talk* to a networked system using GIS and BIM-compliant protocols. This choice is quite arbitrary, as ideally any other geosynthetic can be turn into a smart geotechnical object. In fact, examples of such attempts involving sensor-enable geosynthetics are already well underway in the technical community (Cui, 2018) (Yazdani & Hatami, 2016). One distinctive aspect of this research is represented by the development of a novel data management framework, exemplified on a simple real-case scenario, in which data standardization (GIS/BIM compliance) and data value extraction (via daisy-chained edge and cloud computing) play a central role.

2. Data sources

In the AEC industry data sources are usually related to the so-called “Smart” devices. Data can be gathered by all sort of stand-alone or embedded sensors and most of such sensors convey data that is useful for the property/asset management throughout the life of a building or an infrastructure.

In the geotechnical engineering practice data are usually acquired before or during construction activities and, as opposite to other disciplines of the industry, little to no further monitoring is used to provide geotechnical information throughout the life of the building/infrastructure.

This chapter focuses on the general aspects of geotechnical information management and on the possibilities offered by the *smart device logic*. It starts from the standard engineering practice, that entails well-established in situ and laboratory testing procedures as well as purpose-specific monitoring (e.g. inclinometers, piezometers, etc.). Then it explores the possibility of more long-lasting data sources that mimic the anticipated smart device logic. Finally, it introduces the concept of sensor-enabled geosynthetics as one promising way to fulfill a permanent geotechnical monitoring framework, suitable for both buildings and infrastructure management.

2.1 Typical geotechnical data sources

In common geotechnical engineering practice, data are usually acquired via essentially three classes of methods:

- Direct testing and monitoring: this include drilling and continuous coring, CPT soundings (Figure 4), in-hole installations such as inclinometers and piezometers, direct compaction testing, etc.

- Indirect testing: this includes a wide range of geophysical methods such as GPR (Figure 5), active and passive seismic methods, ERT, non-invasive compaction testing, etc.
- Laboratory testing: i.e. all standardized testing carried out in the geotechnical laboratory to complement the two previous categories.

The majority of such information are, in most cases, gathered at the beginning of the construction project and are at the basis of standard geotechnical reports. Investigation and preliminary monitoring results are then made available to the geotechnical engineers for the further design phases.

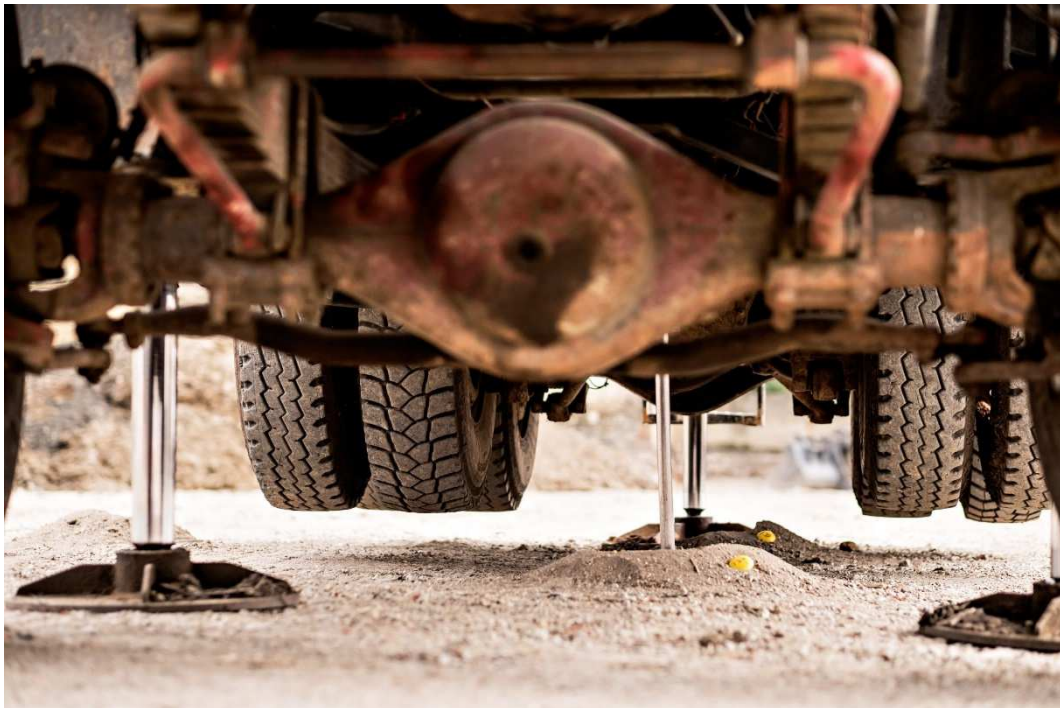


Figure 4 - Typical CPT probing equipment



Figure 5 - Typical GPR equipment

In some scenarios monitoring during construction plays a particularly important role. This is the case of most deep excavation and tunneling works, where systematic monitoring during construction is performed to mitigate the most prominent risks related to ground settlement, collapsing of excavation supports, etc.

Geotechnical information from site investigations, as anticipated, are usually encapsulated in static reports, while monitoring data are usually managed in proprietary servers (typically supplied by the monitoring system manufacturers) and made available to the engineers via dedicated (proprietary) software or web platforms.

Some advanced software systems (an example in Figure 6) allow for an integral visualization of ground investigation and monitoring, most of them leveraging the GIS capabilities to correlate data in a geospatial fashion.



Figure 6 - Commercial software for geotechnical data management through a GIS-like approach

This software, as a necessity, relies on some sort of data standardization to manipulate and exchange information from different data sources. All the commercial packages currently available on the market and capable of this sort of integration rely on the AGS data format to provide a robust data exchange framework (see section 3.4 for further details about geotechnical and geo-environmental data exchange formats).

2.2 “Smart” objects vs. regular geotechnical monitoring

By looking at some developing practices in the field of data integration at the building/infrastructure level, i.e. energy management (Wang, 2013) or OHS (Riaz, 2017) as well as some remarkable research and development works in the geotechnical field (Xue W., 2015), one may become aware of the potential of “smart objects” (i.e. sensor-enabled) when compared with state-of-the-art geotechnical monitoring.

While regular monitoring is usually stand-alone and installed *a posteriori* and might generate “unwanted” costs and interferences on site, sensor-enabled geotechnical objects would be ideally much more site-tolerant and cost-effective. By embedding sensors in e.g. piles, Diaphragm walls, geosynthetic layers/objects, etc. one may be able to collect virtually any kind of valuable information depending on the specific site context without the costs associated to post installation.

The downside of embedding sensors *a priori* is that some high-precision measures would be precluded due to the less accurate installation procedures as well as the potential damages during construction.

Definitely, one may think of smart geotechnical objects as valuable sources of information in situations where monitoring granularity, persistence and repeatability is deemed interesting for asset management purposes and as a valid complement to standard, well-established monitoring techniques.

In light of the considerations above, a choice has been made to explore the opportunity of sensor-enabled geosynthetics to develop the IoT framework based on a plausible use-case scenario.

2.3 Sensor-enabled geosynthetics

Finding a suitable way for extracting information from sensor-enabled geosynthetics is not a novel idea (Yazdani & Hatami, 2016; Cui, 2018),. Most of the researchers in this niche focus on strain monitoring of the geosynthetic layer itself for specific SHM, while, at present, apparently no effort is dedicated to the opportunity of embedding general-purpose sensors with data granularity in mind.

Geosynthetics are a special class of geotechnical objects that are industrially manufactured and, consequently, utterly standardized. Elements such as geogrids (Figure 7), geocells (Figure 8), geo-membranes, geo-composites, geofoam, etc. are used in many different circumstances to build an equally large number of different items. Such items include:

- Road and railway embankments
- Dikes and earth dams
- Stiffened rafts (e.g. for structures laying on ground improved with stone columns, etc.)
- Earth walls
- Landfill capping
- Channel protection surfaces
- Stabilization of erodible slopes



Figure 7 - Geogrids used for soil improvement



Figure 8 - Geocells used for fast access road building

The scope of this work, while keeping the centrality of the IoT logic, is to explore the possibility of natively sensor-enabled geosynthetics that could convey general-purpose information according to the specific use case and site conditions. The prototype presented in this work, that will be described the following sections, includes inertial (acceleration) measurement from sensor-enabled geocells, managed through an IoT framework based on protocols that are, by design and to an adequate extent, compatible with GIS and BIM.

2.4 Smart geocell prototype: physical components

The smart geocell prototype is built on top of a standard, commercially available polyethylene geocell and via consumer-grade electronics. Although perfectible, the setup can clarify the most distinctive features and capabilities of a sensor-enabled geosynthetic layer to be potentially embedded in regular civil engineering applications. The following sections illustrate the main physical components of the system together with some practical considerations based solely on lab-scale applications and testing.

2.4.1 *Sensors*

The sensors used in the prototype setup are consumer-grade triaxial accelerometers. It is a 3V, 14-bit ADC accelerometer capable of handling maximum accelerations in the ranges $\pm 2g$ - $\pm 8g$. In the standard setup (with the measuring range set to $\pm 2g$) the sensor's sensitivity is declared as high as 4096 counts/g. It communicates via the standard I2C port with a Raspberry Pi 3 single-board computer (see fig. 10). In the lab-scale prototype either a single sensor and a dual sensor sensors setup over the same board (see fig. 9) has been used. The full datasheets of the sensor a the controller are provided in Annex 1.

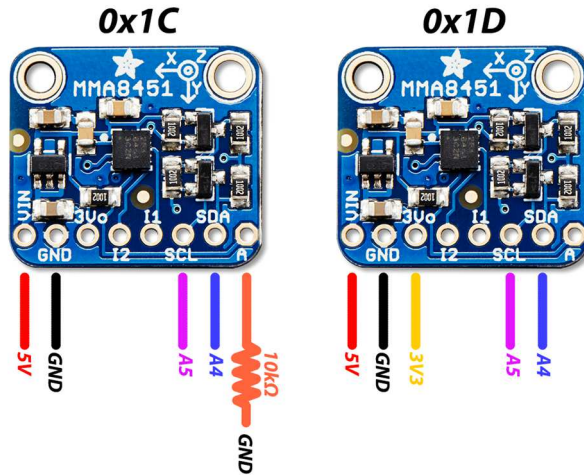


Figure 9 - MMA8451Q 3-Axis 14-bit/8-bit Digital Accelerometer (with I2C wiring schema)

The sensors are used to infer both the variation in tilt and, more generally, to extract information from vibrations induced by external sources. For more details about the basic edge-computing strategies associated with this simple setup, refer to section 3.4.

The sensing system can be further complemented with useful side hardware, such as temperature sensors, an on-board GPS module and RFIDs.

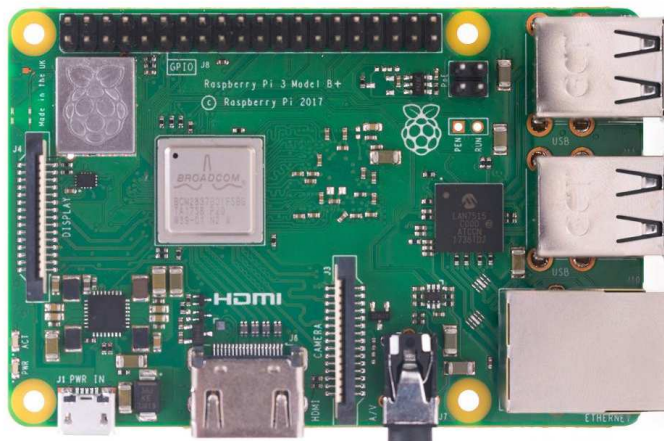


Figure 10 - Raspberry Pi 3 B+ single-board computer

2.4.2 Wiring, enclosure and installation

In order to ensure a decent protection for both sensor and connectors, a simple and effective wiring and enclosure has been developed. A small IP65 split-type enclosure has been adapted to host a single accelerometer, in order to prevent dust and moisture from the fill aggregate particles to get in contact with the electronic component. The enclosure has been solidarized with the geocell sidewall and the outgoing wiring has been performed with an adequately strong ensemble cable.

Ideally, an industrially manufactured smart geocell would include “sensors cells” with built-in wiring to minimize or exclude *a posteriori* installation of the sensing devices. This would undoubtedly ensure faster and more controlled installation procedures and an overall better quality of the enclosure and wiring apparatus.

Finally, considering that especially the sensor orientation might be affected by the geocell installation process, the edge-computing interpretation procedures are tailored to handle this uncertainty by different normalization strategies (see par. 3.4.1). More generally, one should always consider that sensors installed *a priori* on construction materials and elements may suffer from displacement during transportation and installation and the reliance upon precise positioning and orientation, at least outside reasonable limits, should be avoided by design.

2.4.3 Powering and networking

In the lab setup both power and networking are provided through landlines. Nonetheless the very same system could manage more field-tailored power sources (e.g. solar + battery) and can communicate wirelessly via Bluetooth and/or WLAN.

The I2C bus is easily set up through a few lines of (Python) code. The code snippets relevant to the hardware connection and initialization via I2C are reported and commented in Annex 2.

No further study nor prototyping has been dedicated to data transmission technologies specifically tailored to IoT (e.g. LoRa or LPN).

3. Data management

Data management is critical for distributed sensors and IoT applications. This chapter outlines the most promising approaches tailored to geotechnical-related data sources based on GIS and BIM ecosystems and centered onto robust standardization protocols.

3.1 The pitfall of dispersive data

Having the ability to gather and draw to the network large amount of data does not necessarily means extracting most of their intrinsic value from them. Dispersive data is one major pitfall of geotechnical engineering practice, at least in the writer's experience.

In the future perspective of ever increasing data sources, hopefully including sensor-enabled geosynthetics, a dispersive handling is not an option, as it would deprive the IoT item from its additional value and would simply push it out from the large-scale markets of e.g. smart cities and smart infrastructures.

As part of the R&D, two main strategies have been developed to try and mitigate this tendency by encapsulating data from geotechnical investigation and geotechnical monitoring into two complementary ecosystems, GIS and BIM, as already anticipated in section 1.4 and 1.5.

Both strategies aim to centralize data management and visualization while keeping adequate de-centralization for data processing and data transaction (see Chapter 5).

The "GIS approach" that has been developed is quite general and strictly data-oriented and can be applied successfully to structural and infrastructural projects of all sizes (see. section 3.2). On the other hand, the proposed "BIM approach" is more

specific and modeling-oriented and it is more suitable for small projects or subsets of large projects (at least at the present development level of commercial BIM authoring tools), especially where geotechnical issues represent important driving factors (see section 3.3).

Both approaches have been voluntarily developed on top of open standards and platforms, in order to ensure the widest possible interoperability.

3.2 The GIS approach

GIS is naturally a large-scale tool. Its DB structure is capable of handling large amounts of geographical data with a low level of 2D geometrical detail (namely points, lines and/or polygons). This feature, when looking at IoT applications, is useful in infrastructural projects, together with the ability to manage a wide variety of data sources and formats (Li W., 2020).

During the prototyping phases, a standardized data exchange format based on the JSON format and built according to the hierarchical logic of AGS format has been developed. This simple format can transport basic geo-localization information together with a number of custom fields for status, tags, alerts, time series related/derived parameters, etc.

The JSON standard was chosen simply because it is easier to handle within the current GIS software framework, using Python as main programming language and PostgreSQL as DBMS. The custom JSON format is built on top of the AGS4 NZ v1.0.1 standard, that includes appropriate tags for general-purpose monitoring devices (“MONG” and “MOND” headings). Such schema should be fully compatible with the new version of AGS (4.1), released in December 2020, as well as with the new AGSi format (interpreted data AGS counterpart, released in November 2020 in beta

version), which is natively JSON-formatted. The general JSON schema for a single triaxial accelerometer is reported in Annex 2 and it is briefly outlined in Table 2 and Table 3. A vis-à-vis conversion to IFC4 standard has been attempted even though, at least at present stage, AGS format appears superior in terms of standardization capabilities of geotechnically-oriented sensors.

Monitoring Installations and Instruments			
AGS heading	IFC rel. entity	Type	Description
LOCA_ID	IFCSite	string	Location identifier
MONG_ID	IFCSensor	string	Monitoring point reference
MONG_DATE	IFCPropertySet	date string	Installation date
MONG_TYPE	IFCSensorType	string	Instrument type (according to standard abbreviations list)
MONG_DETL	IFCPropertySet	string	Details of instrument
MONG_BRGA	IfcTimeSeries/ IfcPlaneAngleMeasure	number	Bearing of monitoring axis A
MONG_BRGB	IfcTimeSeries/ IfcPlaneAngleMeasure	number	Bearing of monitoring axis B
MONG_BRGC	IfcTimeSeries/ IfcPlaneAngleMeasure	number	Bearing of monitoring axis C
MONG_INCA	IFCTimeSeries/ IFCReal	number	Inclination of instrument axis A
MONG_INCB	IFCTimeSeries/ IFCReal	number	Inclination of instrument axis B
MONG_INCC	IFCTimeSeries/ IFCReal	number	Inclination of instrument axis C
MONG_REM	IFCPropertySet	string	Remarks
MONG_CONT	IFCPropertySet	string	Contractor who installed the instrument
<i>MOND [...]</i>	<i>See Table 3</i>		

Table 2 - AGS-compatible headers and IFC4 compatibility schema for the AGS-JSON format. MONG group headers

Monitoring Readings			
Heading	IFC rel. entity	Type	Description
MOND_DTIM	IFCTimeSeries/ IFCReal	datetime	Date and time of reading
MOND_TYPE	IFCPropertySet	string	Instrument type (according to standard abbreviations list)
MOND_INST	IFCPropertySet	string	Instrument reference / Serial No.
MOND_RDNG	IFCTimeSeries/ IFCReal and/or IFCTimeSeries/ IFCLabel	-	Readings <i>(IFC has different entries for actual readings and status)</i>
MOND_UNIT	IFCPropertySet	string	Units of reading
MOND_METH	IFCPropertySet	string	Measurement method
MOND_LIM	IFCPropertySet	-	Detection limit
MOND_ULIM	IFCPropertySet	-	Upper detection limit
MOND_CONT	IFCPropertySet	string	Contractor who takes readings
MOND_REM	IFCPropertySet	string	Remarks

Table 3 - AGS-compatible headers and IFC4 compatibility schema for the AGS-JSON format. MOND group headers

By using a specific version of AGS as counterpart, it is possible to define a format that is readily translatable into a broadly recognized standard and, meanwhile, is easily manageable within standard GIS platforms, 60% lighter on average (when compared to the CSV standard AGS format) and still human-readable. It is worth noticing that IFC format is more flexible than AGS (see, for instance, the ubiquity of the container class IFCPropertySet, which is conceived to hold properties within a dynamically extensible property tree), while the latter is surely more detailed and less prone to ambiguity in terms of field-specific dictionaries/definitions.

3.3 The BIM approach

Intuitively, BIM is far more detailed than GIS as for geometrical description of physical entities, hence it is more suitable for smaller, circumscribed projects. When looking at IoT, this means BIM would be the ideal interface for sensor-enabled objects at the scale of buildings or small infrastructural elements/networks.

By keeping the basic JSON structure described in section 3.2 and reported in Annex 2, one can perform a translation from such format to the IFCJSON-4 development format, namely to the IFCSensor class (J. Rio, 2013; Smarsly K., 2016), by means of the JSON schema provided by BuildingSmart. Thanks to this logic a straightforward link between the GIS and BIM models including sensor-enabled geosynthetics has been developed.

One major pitfall about this approach is represented by georeferencing. While in GIS georeferencing is native, in BIM the majority of users tend to omit global CSs references. While custom CSs may be beneficial for a number of practical reasons, still not having an indication about global positioning (e.g. through the provided fields in the IFCSite class) is detrimental to a proper integration of data from geographical-sensitive ecosystems (such as GIS).

This research did not aim to produce a solution to this particular problem because it has been assumed that the state-of-the-art would naturally evolve towards a proper management of geographical positioning for BIM models, hence resolving the issue at a more general level before approaching the market with the framework.

3.4 Smart geocell prototype: edge computing

The raw data acquired from the accelerometers is processed on the edge thanks to the capabilities of the single-board computer within the sensing bundle. Edge processing has solved some relevant issues in terms of data transmission, noise reduction and power consumption and has allowed us to promote a de-centralized logic that has proved to be intriguing, especially when looking at the general problem of IoT data security (see Chapter 5).

The following sections present two of the main edge-computing procedures that has been developed for the smart geocell prototype.

3.4.1 Data normalization and standardization

As anticipated in section 3.2 and 3.3, a protocol to translate the accelerometer time-series into synthetic data to be transported via a compact format, which would be readily translatable into AGS and IFC, is designed. To perform such translation the data must first undergo a normalization process. Several approaches were tested during prototyping and the most promising are listed below:

Approach	Distinctive features	Python library
Tilt analysis	<ul style="list-style-type: none">• Intuitive feature• Fast evaluation over subsets• Useful to trigger alarms	Scipy 1.5+
Power spectral density (PSD)	<ul style="list-style-type: none">• Robust feature• Can be used as a noise estimator• Can be used to estimate RMS	Scipy 1.5+

Principal component analysis (PCA)	<ul style="list-style-type: none"> • Can be used in a pipeline with power spectra • Reduces dimensionality while retaining most of the variance • Facilitates further processing (see section 4.3) 	Scikit-learn 0.23+
------------------------------------	---	--------------------

Table 4 - Libraries used for data normalization and standardization

For tilt angles estimation from acceleration time-series the following formulations are used:

$$\tan \varphi_{yxz} = \frac{a_y}{\sqrt{a_x^2 + a_z^2}}$$

$$\tan \theta_{yxz} = -\frac{a_x}{a_z}$$

where:

a_x, a_y, a_z measured linear accelerations in the x, y and z directions

φ_{yxz} roll angle

θ_{yxz} pitch angle

The angles are calculated in accordance with the R_{yxz} rotation sequence.

The following formulations can be adopted, as an alternative to the previous ones, to reduce the numerical effects of the instability regions (where a_z approaches zero):

$$\tan \varphi_{yxz} = \frac{a_y}{\sqrt{a_x^2 + a_z^2}}$$

$$\tan \theta_{yxz} = -\frac{a_x}{\text{sign}(a_z) \sqrt{a_z^2 + \varepsilon a_y^2}}$$

As for power spectral density calculation, the classical Welch's approach is preferred, as it natively reduces the noise impact:

$$S_x(f) = \frac{1}{K} \sum_{k=1}^K P_k(f)$$

where:

P_k is the modified periodogram obtained from the discrete Fourier transform

K is the number of batches

As for PCA, randomized SVD is chosen primarily for its computational efficiency (Halko, Martinsson, & Tropp, 2011). The PCA procedure is classically formulated as follows:

$$\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where:

\mathbf{A} target matrix of dimension $m \times n$

\mathbf{U} matrix with orthonormal columns of dimension $m \times k$

\mathbf{V} matrix with orthonormal columns of dimension $n \times k$

$\mathbf{\Sigma}$ diagonal (singular values) matrix of dimension $k \times k$

The relevant code snippets are reported in full in Annex 2.

Figure 11 reports an example of PSD for a single acceleration component up to the frequency of 512 Hz evaluated during one of the smart geocell lab trials. The PSD is a simple and yet very powerful tool to evaluate both the noise impacts and the signal

RMS amplitude. It is also a convenient way to compare multiple time-series through e.g. Coherence.

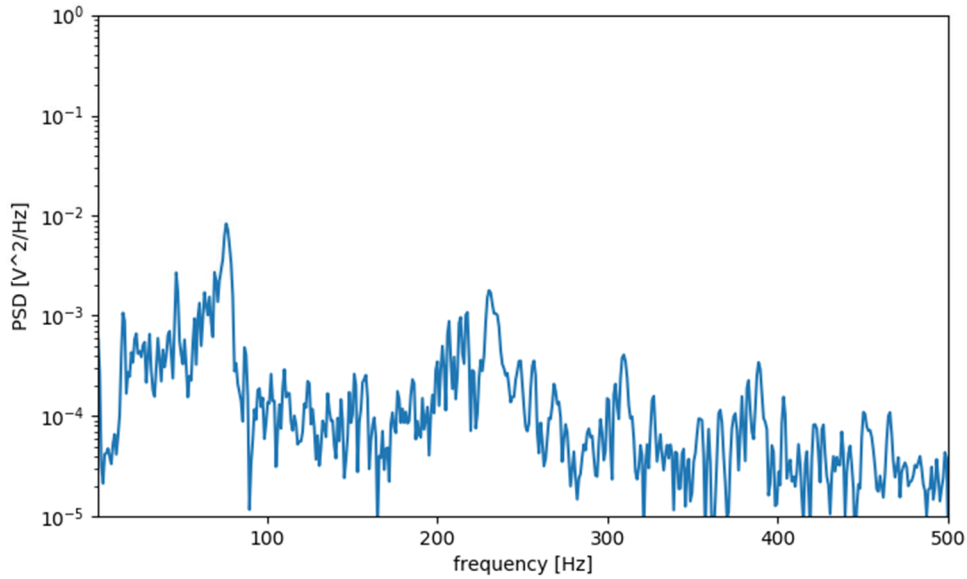


Figure 11 - Example of PSD from the smart geocell single-sensor prototype

Either single time-series noise estimation (average PSD excluding peaks) and RMS (absolute peak of PDS) or multi-series coherence can be used to compress/feature the minimal inertial monitoring information and transport them via the proposed AGS-JSON format. Another type of compression/featuring can be achieved by the proposed PCA algorithm applied to the PSD in a pipeline. Table 5 reports an example of PCA applied to another experimental set, where the PSD variance ratio for the three principal directions was targeted.

Component	Variance Ratio
X	81.0%
Y	14.8%
Z	4.2%

Table 5 - Example of PCA applied to the principal directions of the smart-geocell single-sensor prototype

This form of compression/featuring (together with some other opportunities of the pipeline PSD+PCA, e.g. PCA noise covariance) can be used to prepare the data for further edge and cloud computing. In the proposed example, the X-component carries most of the variance, hence it might be used alone to perform further machine learning while retaining most of the information and, therefore, optimizing the chained procedures.

As anticipated, after the normalization process described above, one can try and translate the compact output in terms of peaks, PSD and/or PCA+PSD in a format compatible with AGS and, potentially, IFC exchange formats (namely IFCJSON formats).

To do so one may take advantage of e.g. MONG/MOND classes of AGS4 format (see Tables 2 and 3), as described in Aguilar (2017). Those classes are, in fact, sufficiently general to be able to host any sort of monitoring data, including the verbose description of data sources and data processing. The same is true for the IFC4 format that includes the IFCSensor object type (Wang, 2013), though such format is still not as mature as AGS for this particular task.

Once the data has been translated into one or both the interoperable formats cited above, the information can be exchanged through any compatible commercial and open source GIS/BIM platforms, both via desktop software and PaaS/SaaS products.

3.4.2 Low-level thresholds handling

Edge-computing can also be used to trigger alarms based on low-complexity procedures. The first and simpler attempt was to set up an alarm triggering criterion based on fixed thresholds and windowed signals. Alarms have been set both on acceleration and tilt, the latter being the most significant for identifying slow and progressive stability/distortion problems. Another proposed, slightly more complex strategy entails an adaptive threshold logic based on a simple unsupervised learning algorithm, i.e. K-Means:

$$\sum_{i=0}^n \min(\|x_i - \mu_j\|^2) \quad \mu_j \in C$$

where:

n is the number of samples

x are the sample feature values (e.g. RMS amplitude, roll, pitch, PCA variance ratio, ecc...)

μ are the cluster centroid feature values

C are the disjoint clusters

The relevant code snippets for low-level threshold handling are reported in full in Annex 2.

Even in this case one may take advantage of the AGS and/or IFC standards to convey the status information from the edge to a GIS/BIM server directly, without the need of full timeseries transfer (see Figure 12 for an example of such logic).

In fact, this could represent the most compact data exchange modality, even more compact than that described in section 3.4.1. As the benefits of data compactness in

a potentially overcrowded IoT environment are evident, one may think of the status-only mode as a normal operation protocol, while retaining the possibility of acquiring the descriptive parameters (or even streaming live data) under certain non-standard circumstances.

```

{
  "PROJ": {
    "PROJ_ID": "4-00495",
    "PROJ_NAME": "Smart_Foundation",
    "PROJ_CLNT": "Biax",
    "PROJ_CONT": "Cresco Group",
    "PROJ_ENG": "ENVIA"
  }
  "LOCA": {
    "LOCA_ID": "ACC1",
    "LOCA_TYPE": "Instrument",
    "LOCA_STAT": "Monitoring",
    "LOCA_LAT": 4929412,
    "LOCA_LON": 686687,
    "LOCA_GL": 0.000,
    "LOCA_LLZ": "EPSG:32632",
    "LOCA_REM": "Smart_Foundation_Location"
  }
  "MONG": {
    "MONG_ID" : "ACC1",
    "MONG_DIS" : 0,
    "MONG_DATE" : 01/01/2020,
    "MONG_TYPE" : "ACM",
    "MONG_CONT" : "ENVIA",
    "MOND" : {
      "MOND_DTIM" : ["04/01/2021 11:45:00", "04/01/2021
12:00:00", "05/01/2021 11:45:00", "06/01/2021 11:45:00",
"07/01/2021 11:45:00", "08/01/2021 11:45:00", "09/01/2021
11:45:00", "10/01/2021 11:45:00", "11/01/2021 11:45:00",
"12/01/2021 11:45:00"],
      "MOND_RDNG" : ["OK", "OK", "OK", "OK", "OK", "OK",
"Tilt Alarm", "Tilt Alarm", "Tilt Alarm", "Tilt Alarm"],
      "MOND_TYPE" : "STA"
    }
  }
}
}
}
}

```

Figure 12 - AGS-JSON for a simple, single-sensor configuration. Status indicator (based on tilt) is stored and exchanged

4. Modelling and analytics

The procedures presented in Ch. 3 and carried out *on the edge*, are meant to provide fast feedback and data wrapping and are not sufficient to extract the most of value from the smart geocell (and from sensors enable geotechnical objects in general).

The following sections present some of the attempts aiming to extend the smart geocell capabilities by further cloud computing. Two main strategies are presented: and they are both discussed in terms of capabilities and implementation on remote computing services.

4.1 Model-based vs. Data-driven approach

As anticipated, several cloud computing procedures that fall into two different strategical categories have been tested. The first strategy, and the most intuitive from an engineering point of view, is founded on the definition of a physically based prediction model that is subsequently fed with data from sensors to trace and even trigger events. On the other hand, the second strategy is based on general-purpose classification/regression models (supervised machine learning models) that are not bound to a specific physical representation and that need to be *trained* before they can be put at use. Both strategies have prediction capabilities over the physical phenomenon of interest, and both have, in the writer's opinion, preferential fields of usage.

The model-based strategy has been deployed in a rather *easy-to-model* scenario involving a raft foundation subject to differential settlements and the data-driven approach (machine learning) in a *hard-to-model* scenario, where smart geocells are allegedly used to detect road/airfield pavement or rail superstructure wearing.

Both approaches proved feasible and showed good adaptivity to the potential and limitation of the simple sensing system. The first approach has been fully developed and it is under testing within a broader commercial initiative. The second is still in its conceptual phase and it has not been fully developed at present.

4.2 Distributed numerical modelling: the smart foundation

The distributed numerical modeling strategy is based on the joint use of a general purpose, open source FEA ecosystem (EDF, 1989-2020) and the cloud computing tools that combine the capabilities of two well known frameworks, namely Docker for containerization/distribution and Kubernetes for instances orchestration.

The first and simplest attempt was part of a broader commercial initiative involving a recently patented ribbed raft foundation system from Cresco Engineers New Zealand (see Figure 13). This system is specifically tailored to withstand the significant distortions caused by expansive or severely liquefiable soils thanks to a peculiar design of the void (plastic) pods.



Figure 13 - Building phase of the voided raft system (courtesy of Cresco Engineering NZ)

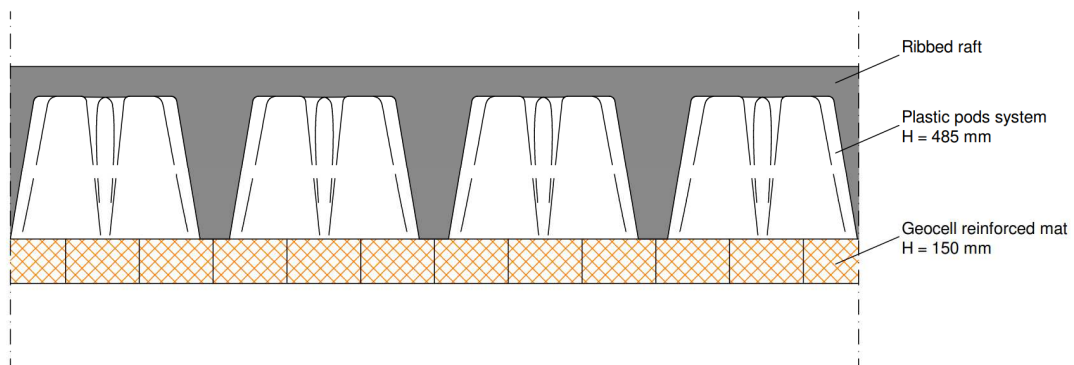
In case of large distortions, the same system can be equipped with innovatively designed perimetral jacking pads that can be used to re-level the superstructure by means of hydraulic jacks (see Figure 14).



Figure 14 - Preparatory phase of the voided raft system: engineered base mat and jacking pads (courtesy of Cresco Engineering NZ)

Thanks to the collaboration with the ribbed raft patent holders and distributors for Australasia, a SaaS platform dynamically linked to Code Aster libraries (EDF, 1989-2020) that is currently used by designers to evaluate the performances and the safety requirements of the novel foundation system subject to different distortion profiles (as anticipated, due to liquefaction and to swelling/shrinkage cycles) has been developed. Provided that geocells have been investigated as a complementary means of differential settlement mitigation to be placed below the said ribbed rafts (see Figure 15), the very same software is being developed further to model the foundation behavior under measured distortions aside from theoretical ones (AS, 2011).

For this particular application, acceleration + tilt for liquefaction and tilt only for swelling/shrinkage are the designated target quantities. In both cases the simple, commercial-grade sensors chosen in the first place for the laboratory setup are potentially capable of handling the target acceleration and tilt ranges (accelerations in the range of 0.1 - 1.0 g and planar rotations in the order of 0.1° or greater).



Sensors location for the model-based approach is quite important, at least in this particular scenario. For the swelling-shrinkage setup, for example, one way to choose the best sensors location under the foundation area would be to use theoretical deformation profiles, like those included in relevant design codes (AS, 2011) and derived from e.g. Mitchell's and Walsh works (Fardipour, 2016; Karunaratne, 2012).

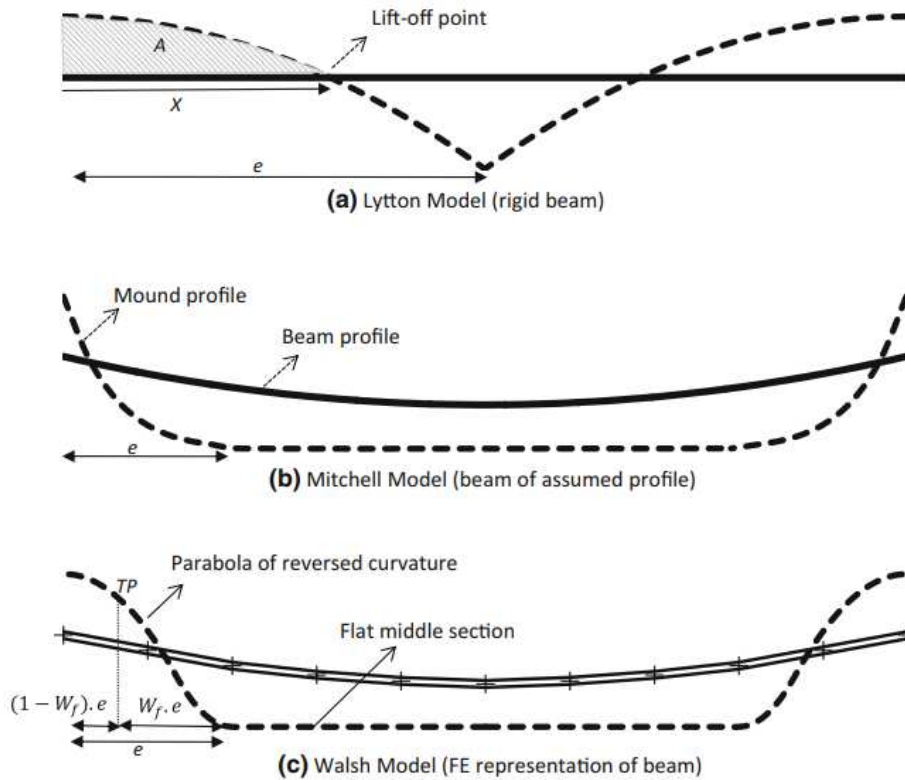


Figure 16 - Typical edge-heave swelling profiles from various authors

Both Mitchell and Walsh models (figures 16.b and 16.c), which are generally more suitable to perform the design of deformable rafts subject to swelling height profiles in the range of 20-100 mm, identify regions of maximum mound curvatures in proximity of the slab edges. This is generally confirmed by field experience. Hence the optimal location of inclination sensors for the swelling-shrinkage setup should be in the within the “active” zone e , close to the edges and possibly in proximity of the occasional critical spots (such as large trees that might influence the suction profiles). Some additional modeling testing will be needed to validate this assumption, as some boundary effects are not considered in the simplified mound-shape methods. Such effects include:

- The stiffening effect of the geocell-reinforced mat
- The effect of the damp-proof membrane
- The effect of the system's voids in terms of pressure localization and swelling relief under the ribbed raft

The first full-scale prototype should then serve as a reference to fine tune the integrated monitoring as well as to calibrate the provisional numerical modeling and geocell-reinforced mat design accordingly.

In this scenario, the distributed numerical modeling based on measure points could be beneficial for progressive product optimization, also taking advantage of spatial analysis of monitoring data based on the GIS logic.

This is especially true for expansive soils. This is, in fact, a largely diffused and frequent phenomenon (much more frequent than severe liquefaction, being related to perpetual seasonal water content variations) which underlying physics is quite complicated and extremely site dependent. By accumulating the right amount of geospatial data, the developers and designers of the system would benefit from e.g. "performance maps" as well as additional design and optimization criteria.

As anticipated, the software (which cannot be reproduced in full in Annex 2 due industrial and commercial secrecy arrangements) runs in the form of a REST API containerized in a Linux-based Docker and orchestrated by Kubernetes. The software logic, involving the anticipated prototype health monitoring strategy, is outlined in Figure 17.

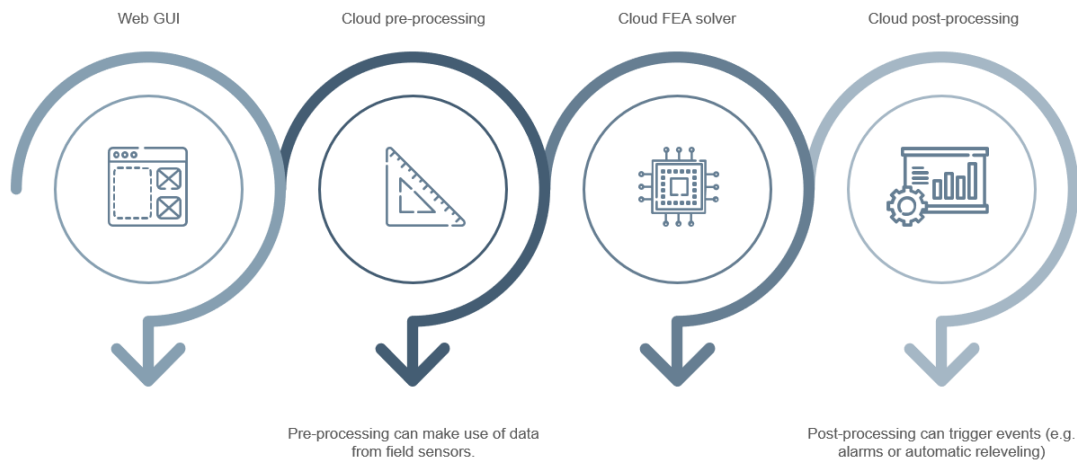


Figure 17 - Basic architecture of the SaaS

The SHM coupling strategy includes, in brief:

- The possibility to use both theoretical and measured distortion profiles as inputs for the modeling
- The possibility to trigger events in real-time based on the modeling outputs. This may include simple alarms and push notifications as well as more complex actions (i.e., hydraulic jacks control and actuation for releveling)

In the case of automatic releveling, the potential benefits of equipping the plastic pods with sensors as well are sought. For this purpose, as the expected strains/rotation of the structural elements would allegedly be smaller than those of the geocell-reinforced mat, one shall explore other sensing options that may prove more suitable than the first prototype setup.

4.3 Distributed Machine Learning

The distributed machine learning approach can be very useful to extend the capability of the collected dataset for both classification and regression (prediction) purposes. This approach, when compared to the model-based approach outlined in the previous section, is not constrained to a pre-determined physical representation of a specific phenomenon. Hence its applicability is virtually far more general and multi-tasking by nature.

The most interesting distributed machine learning procedure for smart geocells, as well as smart geotechnical objects in general, targets the predictive maintenance problem. In fact, this is a very tough problem to be addressed with the model-based approach described in section 4.2, while, with a sufficiently large dataset available, might become easier to solve using supervised learning strategies. This is due mainly to the ability of machine learning procedures to cluster and classify datapoints based on any feature set and even to emphasize relationships that might be invisible or counter-intuitive for a human model builder.

Since the full development of such procedure depends on the availability of large sets of data, it was not possible to test the prototype at scale. Nonetheless, in section 4.4 a brief description of the proposed approach is presented. Such approach is based on state-of-the-art classification/regression supervised learning algorithms including:

- Random Forest (RF)
- Elastic Net

Especially RF has been already experimented for classification in other fields of geotechnics with positive outcomes (Liu, et al., 2020).

4.4 Smart geocell prototype: machine learning

Machine learning, like remote modeling, takes place on cloud servers specifically designed to handle this sort of algorithms. While the most demanding machine learning procedures (e.g., Deep Neural Networks) might require different architectures than those built for the prototype SaaS and outlined in section 4.2, the procedures anticipated in section 4.3 are simple and lightweight enough to run onto the same, low memory single CPU, Docker + Kubernetes architecture. The following sections describe two of the tentative procedures, based on the smart geocell setup, that has been developed for predictive maintenance purposes.

4.4.1 *Status classifier*

For the generic task of defining a “status” of the sensor-enabled geotechnical object, Randomized Decision Trees (supervised) classification algorithm has been tested. The simplest status classes can be set to identify:

- an OK status (no action needed)
- an off-line or no data status
- a pre-alarm status (might be used to trigger continuous data stream to inspect the data in deeper details)
- an alarm status (might be used to trigger some mitigation measures)

Being a general purpose, supervised classification algorithm, Randomized Decision Trees (aka Random Forests, RF in short) can be used virtually in any application context, provided that the collected dataset is sufficiently good to properly train-test it.

By definition, Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal of DTs is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Given training vectors x_i and a label vector y , a decision tree recursively partitions the space such that the samples with the same labels are grouped together.

Let the data at node m be represented by Q . For each candidate split $\theta(j, t_m)$ consisting of a feature j and a threshold t_m , partition the data into $Q_{left}(\theta)$ and $Q_{right}(\theta)$ subsets:

$$Q_{left}(\theta) = (x, y) \mid x_j \leq t_m$$

$$Q_{right}(\theta) = (x, y) \mid x_j > t_m$$

The impurity at m is computed using an impurity function H , the choice of which depends on the task being solved (classification or regression):

$$G(Q, \theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta))$$

If a target is a classification outcome (like in this case), taking on values $0, 1, \dots, k-1$, for node m representing a region R_m with N_m observations, let

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

be the proportion of class k observations in node m .

One of the most versatile impurity function in this context is Gini:

$$H(X_m) = \sum_k p_{mk} (1 - p_{mk})$$

where X_m is the training data in node m .

The purpose of assembling multiple DTs (i.e. implementing the Randomized Decision Trees logic) by inserting some randomness sources (e.g. bootstrapping both samples

and predictions) is to decrease the variance of the estimator. Indeed, individual decision trees typically exhibit high variance and tend to overfit. The injected randomness in forests yield decision trees with a certain level of decoupling of prediction errors. By taking an average of those predictions, some errors can cancel out. Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant, hence yielding an overall better model.

An implementation attempt for RF Status classifier for this framework is reported in Annex 2. This attempt considers a very general purpose for this classifier, making it potentially suitable for the smart foundation prototype, as well as for other, unexplored applications of smart geocells.

4.4.2 Predictive maintenance regressor

Predictive maintenance is a very interesting topic for sensor-enabled geotechnical objects. Especially in infrastructural IoT, a predictive maintenance regressor can prove extremely useful in anticipating damaging and wearing patterns and, ultimately, in optimizing the most delicate maintenance activities.

By implementing the very same logic of predictive maintenance used e.g., in the Oil & Gas industry (Rayes, 2019), one can define a set of (supervised) ML algorithms that can be used as regressors or part of regressors based on time-series.

Furthermore, the potential of the Elastic Net (supervised) regression/regularization algorithm has been tested as well. One of the main advantages of such algorithm is

that it is particularly suitable for sparse datasets, which can be very useful if one needs to keep the data stream as small as possible and still be able to perform a robust enough prediction.

Elastic Net is a linear regression and regularization model that aims to minimize the following loss function:

$$L(\hat{\beta}) = \frac{\sum_{i=1}^n (y_i - x_i \hat{\beta})^2}{2n} + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^m \hat{\beta}_j^2 + \alpha \sum_{j=1}^m |\hat{\beta}_j| \right)$$

where the regularization parameters α and λ are set through cross-validation. An implementation attempt for Elastic Net regressor for this framework is reported in Annex 2

When comparing the status classifier based on RF (section 4.4.1) with the Elastic Net regressor implementation, one shall consider that the latter is conceived to be interoperable inside a broader and multi-purpose infrastructural IoT environment (as a basis for higher-order ML/DL procedures), rather than a stand-alone estimator. One of the main reasons for that choice is that, especially for infrastructures, interoperability of different data sources is deemed crucial to establish data-driven predictive maintenance protocols. As for roads, for example, one could consider the integration of the following data sources:

- fixed sensors (incl. smart geocells)
- mobile (vehicle-mounted) sensors
- atmospheric monitoring data
- InSAR and other satellite imaging

5. Data security

When considering potentially large networks of sensors, data security is one aspect to be studied in depth before getting to production. This is especially true if one considers the intrinsic value of data extracted from widespread monitoring in terms of industrial advantage (Zuboff, 2018).

This chapter analyzes the data security aspects of sensors enabled geosynthetics in view of the latest research about substantial risks of IoT systems (Brous, 2020). Firstly, the potential drawbacks of data centralization and, on the other hand, the benefits of a decentralized, smart geosynthetics network are explored. Secondly, the potential role of cryptography and blockchains to ensure security and trust in a decentralized network is analyzed. Finally, the role of current policies and regulations applicable to IoT as well as data ethics issues are discussed.

5.1 Centralized vs. Decentralized data

According to (Rayes, 2019) a regular IoT architecture is made up of three distinct domains:

- *Sensing domain*: the domain made up of all smart objects that can collect data from the environment.
- *Fog domain*: the domain of fog devices (i.e., gateways) that aggregate sensors data, stores them and perform preliminary operations onto them.
- *Cloud domain*: the domain composed of cloud devices/services (i.e., servers) that orchestrate all fog devices and perform the heavy-computational processes to extract the most of data value.

This hierarchy is fully centralized, being the cloud servers the entities that retain and manage most of the system capabilities and value. This means a full IoT sensors network (e.g., a full stack of smart geocell data points along a railway segment) is potentially vulnerable to a single malfunctioning or security breach on the managing server side.

As clarified in Chapters 3 and 4, a tentative has been done to overcome this security issue by delegating some actions to the sensing domain (through the well-known logic of edge computing). Moreover, an alternative high-level orchestration method, based on distributed ledgers rather than centralized servers has been explored. This particular aspect, which will be described in further details in section 5.2, involves the use of cryptography and blockchains that would provide an immutable ledger for data-driven events (transactions), rendering the IoT process much harder to manipulate and disrupt and, on the other hand, way easier to control and inspect.

5.2 Cryptography and Blockchains

This section introduces a conceptual design for a simple private blockchain implementation tailored to the prototype sensor-enabled geotechnical objects, based on the (open source) Hyperledger Iroha project and the YAC consensus algorithm (Muratov F., 2018).

As anticipated in section 5.1, sensor-enabled geotechnical objects as well as IoT in general, need to account for complex network functional and security design aspects, such as crash fault tolerance, cryptography, etc.

One way to pre-design a system to be able to address such issues in a complex IoT network is to make use of blockchains.

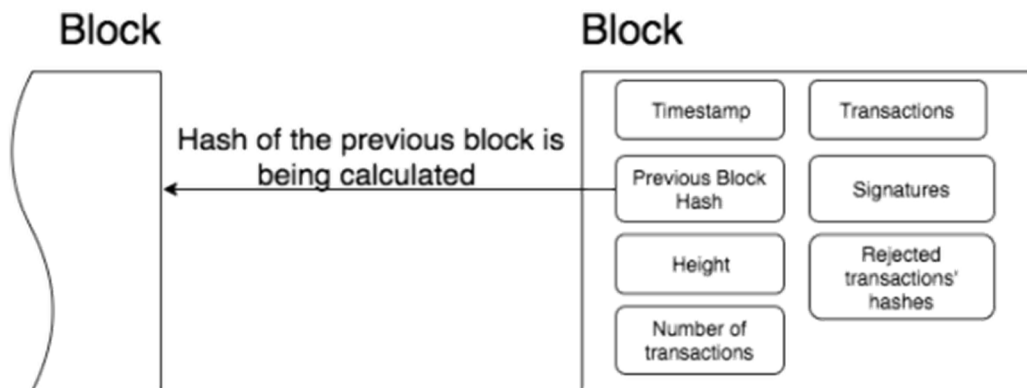


Figure 18 - General concepts of a distributed ledger in blockchain technology

In very simple terms, a blockchain is a time dependent list of records that are linked one another through cryptography. Blockchains can be either public (e.g. Bitcoin or Ethereum) or private. The terms blockchain usually refers to the “distributed ledger” logic, where the blockchain is shared and maintained by a peer-to-peer network with an underlying consensus mechanism (protocol). One distinctive feature of blockchains is their low vulnerability to manipulation, which make them suitable for a range of applications involving trust and data security. Figure 18 briefly outlines a typical blockchain block structure.

In this specific case, where peers are single sensing nodes or single gateways depending on the chosen architecture, the private blockchain structure can be used to:

- Provide an alternative storage means for e.g., some classes of potentially sensible generated (meta)data.
- Prevent data manipulation on the edge (sensing network fault-tolerance).
- Allow for better integration with other IoT systems/networks while keeping the adequate level of data privacy and security.

- Allow for data stream and (read-only) control by some authority which has access to the private blockchain.

In the conceptual design, which is described further in Annex 2, an attempt has been made to deliver an Hyperledger Iroha implementation that exploits its straightforward client-server abstraction for peer network interaction and its distinctive low-latency consensus algorithm. With this initial design, the prototype has been allegedly equipped with all the basic features of a private blockchain, targeting the following possible development aspects (Paik H., 2019):

- Blockchain architecture as a (meta)data store
- Distributed data analytics
- Governance (legal, quality, etc.)

Even though the current prototypes rely strongly upon the standard IoT architecture (see section 5.1) and centralized third-party cloud services providers, it would be desirable to interpose a blockchain layer in between such external entities and data production. This strategy aims to increase the control over the generated data while keeping the benefits of high-end and reliable cloud services provided by e.g., Google Cloud, Microsoft Azure or Amazon AWS to cite a few.

5.3 Policies and regulations

This section briefly outlines some of the most relevant policies and regulations that may apply to and interact with plausible further developments for the sensor-enabled geotechnical objects.

First of all, one needs to consider the relatively broad list of industry organizations that are currently involved in the process of standardizing the IoT ecosystem. Just to cite a few: IEEE, IETF, IPSO Alliance, LoRa Alliance, etc.

All such organizations are targeting different technical aspects of IoT in order to construct a robust policies architecture on top of which virtually any IoT system manufacturer can build its own products, still no organic framework is available to date (Rayes, 2019).

It is the writer opinion that those organizations should be able to deliver, in reasonable a short period, an interoperable and secure environment that can host IoT products, especially those built on top of open-source solutions.

Provided that technical policies are still underway, no comprehensive regulatory framework should be expected very soon, even though the Regulation 2016/679 (GDPR) and the Regulation 2018/1807 are setting out a general legal path for the IoT and “data market”, at least in the EU (Boardman R., 2019).

One interesting technical regulatory approach is provided by the EN ISO 19650 series of standards. These standards are strictly related with the BIM regulatory framework and they deal primarily with information management. One of the main scopes of such standards is to set out *efficient and effective transfers of information between those involved in each part of the life cycle – particularly within projects and between project delivery and asset operation*. This scope is part of a broader commitment to information quality within the BIM ecosystem, involving a number of other standards as illustrated in figure 19.



Key

AIM Asset Information Model

PIM Project Information Model

A Start of delivery phase (see 3.2.11) - transfer of relevant information from AIM to PIM

B Progressive development of the design intent model into the virtual construction model (see 3.3.10 Note 1)

C End of delivery phase - transfer of relevant information from PIM to AIM

Figure 19 - Standards framework including EN ISO 19650

The EN ISO 19650 promote general, rather “philosophical” requirements rather than well-determined protocols, hence the leave space for a certain customization in the BIM area (for example, refer to clause 5.6.2 defining the Common Data Environment logics).

Needless to say that data standardization and interoperability, as e.g. conceived for the prototype (see section 3.4.1), should be always pivotal while defining a link between a sensor-enabled object and its digital twin inside the BIM model, especially for the asset management perspective.

5.4 Data ethics

When dealing with data mining and data value extraction, one must be aware of the underlying data ethics issues. According to (Floridi, 2016), *social preferability* should represent a key principle in data science and all project relying on big data and AI to produce value should account for potential positive and negative social impacts.

In general terms, the present project should have limited design social impact and would rather involve only the specific domain and stakeholders. More generally, although the benefits of distributed monitoring are quite clear, one might argue about the potential ethical problems, such as unwanted and/or possibly damaging use of macro-data.

The prototype has been designed with particular attention to this very aspect by pre-designing a security layer in the data management pipeline based on the blockchain logic. Such layer would provide, at least to a certain extent, a way to establish some of the fundamental pillars of data ethics, such as:

- Transparency
- Accountability

The very same blockchain layer could be used to facilitate a dynamic consent logic for e.g., the premise owner, even though this topic lays far outside the particular scope of this work.

Even though the prototype deals with these ethical issues only preliminarily, it is reckoned that they will be crucial for the future work on the sensor-enabled geotechnical objects, as well as for the IoT industry in general.

6. Economical considerations

Both hardware and software (SaaS) prototypes have been investigated from the economical and business development perspective. This chapter describes the ideal contexts of application of the prototype system and exposes some cost/benefit considerations regarding the use of smart geotechnical objects in AEC industry.

6.1 The general paradigm of data value

In Rayes (2019) the concept of *anything as a service* is introduced to emphasize the importance of the data-driven business model in current times. In the AEC industry the most known example of such paradigm is the Smart Building IoT model coupled with BIM platforms.

Provided that BIM and IoT are progressively extending to the infrastructural domain, the potential value of geological and geotechnical data (including data collected from smart geotechnical objects) is becoming visible.

One distinctive aspect of the data value paradigm, considering the *anything as a service* statement, is that one might look freely at all the possible use cases of the collected information with a service in mind. In the case of data collected from geotechnical objects this translates into considering value not only from the health monitoring perspective (which is undoubtedly a primary goal), but also for the sake of a number of different stakeholders.

In this specific case both geocells and rib-raft foundation system manufacturers were interested in the potential usage of the monitoring data to track their products performances throughout their lifespan and ultimately to optimize their design, and also as a commercial aid to support their businesses.

In the following sections both costs and benefits of geotechnical data mining related to sensor-enabled geotechnical objects are evaluated, based on the direct experience with the hardware and software prototypes and products presented in this research.

6.2 The costs of geotechnical data mining

The costs of data mining in the geotechnical domain is quite variable depending on a number of factors, related to sensing technologies (MEMS, fiber optics, vibrating wire, etc.), installation procedures (surface, embedded, in hole, etc.) and data collection and management. In this specific case study, the following costs related to the hardware and software sub-products can be identified:

- 1 Naked hardware costs, including micro-computer unit(s), board side components and MEMS sensors.
- 2 Wiring and enclosures
- 3 Powering and Networking
- 4 Naked software costs, including edge and cloud computing procedures
- 5 Cloud computing hosting and data storage costs

Thanks to the extremely affordable solutions for good quality micro-computers and MEMS sensors available on the market, point (1) proved to have a reasonably low impact on the full system costs, rendering such system allegedly sustainable for massive production. Point (2) has not been explored in view of industrialized production, which might include natively wired geocells for the most beneficial solution. On the other hand the lab test prototype of smart geocell was equipped with off-the-shelf components for wiring and enclosure, which proved fairly adequate and cheap to buy. Point (3) really depends on the installation context. For the smart

foundation scenario this cost is virtually very low, as one may take advantage of on-premises power grid and internet connection with very limited impact in terms of installation and running costs. For the infrastructural scenario one may still consider acceptable costs related to massive installations, provided that infrastructural IoT in general, at a certain point, would ensure a networked ecosystem characterized by a simple tap-in logic.

As for point (4), this represents a major upfront cost for specifically tailored systems. Since the present work was mainly focused on this aspect of production, the cost analysis is hopefully more detailed and useful on this very topic. A simple cost breakdown structure for the software development is presented in the table below.

Project Stages	Share of overall budget	w. Time (hrs.)	Cost (EUR)
Learning	4.8%	50	1,500
Planning and documentation	7.2%	50	2,250
Development	43.2%	300	13,500
Testing	14.4%	100	4,500
Scope changes/Contingencies	21.6%	150	6,750
Cloud Services (testing phase)	1.6%	-	500
Integration within the general SW framework	7.2%	50	2,250
Total cost to market		700	31,250

Table 6 - Internal costs to market for the SaaS

The software development process was basically spiral (O'Regan, 2017) and was characterized by many subsequent modification, (re)prototyping and validation phases. This is quite natural when dealing with an highly specialized, tailor-made

software that really took form during the process and was finally independently peer-reviewed.

As for running costs, one distinctive aspect of the project was that a large use of open-source platforms has been made, remarkably also for the most sensitive FEA and ML code blocks (based on Code-Aster and Scikit-Learn). Provided that both platforms proved to be extremely reliable and suitable for this form of development, it was possible to obtain a fully functional SaaS that did not depend on any licensed software, hence reducing the software upfront at minimum.

Finally, looking at point (5), both the custom made and the commercial grade solutions from well established cloud service providers (e.g., Google, Microsoft, etc.) have been explored, the latter proving the most economical for the particular level of development. This thanks to extremely low upfront costs as well as easily scalable solutions, together with the possibility to deliver the SaaS in different regions (in this case Australia and New Zealand with potential extension to India) with a very low risk of service disruption.

In a generalization attempt based on this project's particular case, one may conclude that the costs associated to sensor-enabled geotechnical objects rest mainly on the software development side and, more broadly, on the processes that generate value from the data extracted in this non-conventional monitoring scenario. This is quite understandable, considering that consumer grade electronics rather than custom one can perform adequately, while software development needs field-specificity to deliver something valuable.

6.3 The benefits of geotechnical data mining

As any other data source, sensors enabled geosynthetics and, more generally, geotechnical monitoring data, can generate value according to its specific nature. Provided that the vast majority of geotechnical data is represented by time series collected in a specific fixed location, its value is directly derived from its *contextual* as well as *behavioral* attributes (Aggarwal, 2015). Either temporal and spatial information is vital and may store high intrinsic value. For the particular smart foundation system applied to the swelling soil context, value may be sought in e.g.:

- Seasonal variation of maximum swelling/shrinkage edge tilt
- Spatial correlation between foundation distortion and e.g., climatic zone, rainfall patterns, etc.

From a commercial point of view, it is important to notice that data value cannot be measured *per se*, rather it can be evaluated in relation to the potential competitive advantage it might generate once collected.

In the case of this research the most immediate competitive advantage involves the designers and manufacturers of the system. In fact, they showed interest in using the data findings to optimize the foundation system and to provide a clear means of product evaluation for their current and potential customers. They may also look at different use cases by, e.g., simulating engineering scenarios that are not included in the relevant design codes, based on actually measured data. Another interesting potential data value source is represented by the integrability of inertial monitoring with broader IoT ensembles. For instance, one may look at ground-coupled (base) accelerometers as part of more complex IoT devices for dynamic-based SHM, vibration control, large-scale seismic monitoring and early warning networks, etc. In this scenario, the designer and the manufacturers of such sensor-enabled

geotechnical system may benefit from extra competitive advantage, especially in high seismic risk areas.



Figure 20 - Business development strategies around the R&D

Figure 20 outlines some of the possible business development strategies based on data mining and value. It is worth noticing that, aside of these far-reaching potential growth steps, one might also look at cost reduction as the first, more immediate goal that can be reached through data mining.

In this specific use case, as anticipated, one potential optimization of the smart foundation system based on mined data could lead to a progressive reduction in on-site building costs, namely reinforced concrete and geocells related costs. This cost reduction strategy, that would probably appear slow at the beginning, would allegedly have a progressively increasing impact while the designers and the manufacturers move from small-scale (typically single housing at present) to larger scale scenarios. Finally, from the business point of view and as per the initial declaration of this thesis, the main goal was to develop a general framework that would be capable of handling geotechnical monitoring data, exchanging them via GIS/BIM compatible data formats,

building value from them via model-based and/or ML strategies and finally wrap such value as tailor-made software products through the SaaS/PaaS logic.

This prototype frame presented in this research, developed during a 3-year R&D process, would be at the very base of further R&D and it is currently part of several commercial initiatives involving EPC and asset management clients.

7. Concluding remarks

7.1 Review of the research

This research aimed to define a general framework for sensor-enabled geotechnical objects that was suitable to host a smart geosynthetic prototype. Such framework has been developed based on two main data management ecosystems, namely those typical of Geographical Information System (GIS) and Building Information Modeling (BIM). One first effort was dedicated to the definition of a general data exchange format that could render the generic sensor data readable and manageable for the majority of commercial as well as open-source GIS and BIM platforms. It was found that the most promising way of achieving such standardization was to make use of the existing Association of Geotechnical and Geoenvironmental Specialists (AGS) data exchange format, which has been adapted in a novel AGS-JSON format to ease the further use in relational databases and to facilitate the vis-à-vis conversion in and from the Industry Foundation Classes format (IFC).

Following the generic data format adoption, testing and adaptation process, a prototype of sensor-enabled geosynthetic has been developed on top of a standard geocell product and making use of consumer-grade electronics. A series of “smart” geocell prototypes has been developed with both single and dual-sensors configuration for each cellular element. It was found that, at least for the controlled laboratory conditions, a very simple and cheap arrangement in terms of enclosure and wiring was sufficient to ensure the proper functioning of the sensors in the alleged less-demanding working conditions.

The edge computing software attached to the smart geocell prototype has been developed with the primary aim of reducing the amount of data to be exchanged on

the network while retaining most of its intrinsic and predictive value. To attain this goal, two edge computing procedures has been developed. The first was a simple preparatory signal analysis tool, capable of extracting tilt angles form the accelerometer data for further, FEA model-based cloud computing. The second was studied to compact the time series data by a classical frequency-domain conversion through the computation of power spectral density (PSD). Both tilt angles and PSD-compressed time series could be easily stored, exchanged and managed through lightweight AGS-JSON files. Another procedure, based on the well-known unsupervised machine learning K-Means, has been proposed to attain a robust and adaptive alarm-triggering.

As for cloud computing, firstly an adaptation of a formerly released SaaS has been developed to solve a FEA model based on both theoretical and measured distortion profiles, the latter being based on tilt angles estimations. This was part of a broader commercial initiative involving a recently patented voided raft modular system coupled with a geosynthetic-reinforced base. The software was designed to model the two most demanding working scenario of such system, namely expansive soils and liquefiable soils. Furthermore, a less in-depth effort has been dedicated to exploring the potential of general-purpose machine learning procedures for diverse scenarios. It was found that even light supervised learning algorithms (including Random Forest for classification and Elastic Net for regression) could provide a sufficiently adaptive framework for mass data normalization. Such normalization is thought to be preparatory, targeting a broader IoT infrastructure.

Considering the potential pitfalls of smart geotechnical objects immerse in large IoT ecosystems, after facing data compactness and computational efficiency, a very preliminary data security protocol based on the distributed ledger technology (blockchain) has been prototyped. Such protocol is based on the Hyperledger Iroha

framework and its most evident role would be to provide an immutable record of “data transactions” among different nodes in the same IoT network. Other security-related benefits of the adoption of a distributed ledger linking the sensing nodes/gateways would include secure storage for small amounts of sensible (meta)data, increase in the sensing network fault-tolerance, increase in scalability and transparency of the IoT network. This last software development effort has been dedicated to align the potential products of this research to the most important trends in terms of data management and safety, including those circumscribing the BIM workflow.

Finally, a brief cost vs. benefit analysis has been carried out based largely on the smart geocell laboratory prototype and on the commercial initiative involving the aforementioned SaaS, showing that a smart geosynthetic solution entailing off-the-shelf electronics and industry standard cloud computing is practically viable and sufficiently scalable from the economical point of view.

7.2 Discussion

In this research the standardization of geotechnical monitoring data represents a central topic and it clearly has had a pivotal role in all the strategical choices made along the way. The adoption and tailoring of the AGS data exchange format have proven a solid way to construct a link between different industry standards while keeping the adequate flexibility for the smart geocell prototype. It is the writer’s opinion that the ability to exchange and manipulate monitoring data throughout different operative ecosystems (e.g. GIS and BIM) dramatically increases the potential of any monitoring device, including the one developed in this work. Moreover, the attempts to reduce the sensors time-series data in easy-to-transport formats (at different complexity levels) has been carried out precisely to ensure a proper translation

pipeline to the proposed AGS-JSON file format while keeping the network loading at minimum. This strategy may need extra resources on the edge, but it aims to solve the potential bottlenecks of bulk data transmission and server-side continuous elaboration.

Another crucial aspect of our research relates to the dual approach to data value extraction via cloud computing. The so-called “model approach” has been developed within a broader commercial initiative to resolve a specific engineering problem with well-defined boundaries and objectives. It was chosen to develop a FEA code that could seamlessly incorporate sensors data in the form of a SaaS. Such software was built on top of industry-standard open-source frameworks and it is currently under beta testing. This is the most developed part of the work, as the SaaS is fully functional and sufficiently validated at present. The other approach that has been developed at a more rudimentary stage (namely the purely “data-driven approach”) relies upon different supervised learning algorithms that tackle either status classification (adaptive alarm triggering) and regression with a more general perspective for larger IoT use case scenarios. For such specific research aspect, there was no attempt to push the software design further (e.g. for targeting predictive maintenance for infrastructures), as it is expected that more advanced cloud-based data value extraction protocols will benefit from an integration of different data sources, including the one provided by smart geosynthetics.

On the other hand, the sensor-side of the prototyped system has not been tackled as thoroughly as it was desirable in the first place. As a result, the hardware (electronics, enclosures, powering, etc.) and the communication protocols have been developed at a very embryonal level and will surely need serious improvements to be deployable as functioning monitoring nodes in the diverse operative scenarios. In fact, even though the proposed sensor node set-up might suffice for the first scheduled field

application for the smart geocell, it is surely lacking some relevant engineering, especially for off-grid and harsh environmental conditions.

Furthermore, data security has been faced superficially, even though the proposed method based on the distributed ledger logic appears promising and sufficiently easy to implement and scale for the intended application of the smart geocell.

7.3 Future research and developments

At present the prototype presented in this research is deemed sufficient to convey the benefits of sensor-enabled geosynthetics and to captivate the market for further investments. Both the basic sensors setup and the SaaS platform are currently ready to serve the first testing installations for the geocell-enhanced ribbed rafts.

One possible improvement fork for the SaaS would be to include procedure for meta-modeling and uncertainty treatment to be linked with the FEA modeling. Since the initial choice was to utilize the Salome Meca (Code Aster) framework for FEA modeling, it appears natural to make use of the OpenTurns tools for the purpose, as they are already part of the aforementioned open-source initiative by EDF.

Based on the first field tests, further development will be planned to tackle the aspects which are more distinctive for infrastructural applications, starting from the off-grid strategies for powering and networking. Furthermore, based on the developments of the IoT ecosystems for infrastructures, it will be necessary to tailor and/or reformulate the proposed cloud-based procedures in order to optimize the data preparation for the higher-level algorithms.

Lastly, the promising data security solution based on the blockchain technology deserves a dedicated development pipeline, that will allegedly benefit from other

similar attempts from AEC industry as well as other sectors involved in the development of new IoT solutions.

Acknowledgements

I would personally like to acknowledge the importance of the SEHM2 PhD program in the development of this R&D work. Getting in touch with professors and researchers from other engineering disciplines, all working in the different, overlapping fields of monitoring and IoT, was fundamental to develop the broader view required to shape the prototypes in their most critical aspects, including those which are least familiar to us.

I would also like to underline the importance of the work carried out by many other Italian and international research groups, delivering astounding contributions in the fields of BIM and IoT for the AEC industry. A particular mention goes to the FED Spinoff from the Federico II University Naples, whose contribution in the field of geotechnical BIM has been paramount in inspiring and steering the research.

Finally, my special thanks go to Prof. Laura Tonni for her guidance and support, to Prof. Alessandro Marzani and Prof. Tullio Salmon Cinotti for their relentless effort with the SEHM2 PhD program, to Prof. Settimio Ferlisi and Prof. Lucia Simeoni for their thorough and stimulating reviews, to the friends and colleagues from Cresco Engineers New Zealand and their partners and to the friends and colleagues from ENVIA, who fully and actively supported me and my work during this 3-year research project.

References

- Ademci, E. G. (2018). Review of Studies on BIM Adoption in AEC Industry. *Conference: 5th International Project and Construction Management Conference (IPCMC)*, (pp. 1046 -1055). N. Cyprus.
- Aggarwal, C. (2015). *Data Mining: the textbook*. New York: Springer.
- Aguilar, K. e. (2017). *Electronic Transfer of Geotechnical and Geoenvironmental Data - AGS4*. Bromley, Kent: Association of Geotechnical and Geoenvironmental Specialists (UK).
- Ahmed, V. A. (2018). Challenges and drivers for data mining in the AEC sector. *Engineering Construction & Architectural Management*, 25.
- AS. (2011, January 17). AS 2870-2011 Residential slabs and footings. Australia: Standards Australia.
- Bello-Salau, H., Aibinu, A., Onumanyi, A., Onwuka, E., Dukiya, J., & Ohize, H. (2018). New road anomaly detection and characterization algorithm for autonomous vehicles. *Applied Computing and Informatics*, 1-10.
- Berg, C. N. (2015). *Transport policies and development*. World Bank Group.
- Boardman R., M. J. (2019). *Guide to the General Data Protection Regulation*. Bird & Bird.
- Boas, M. (1983). *Mathematical Methods in the Physical Sciences*. Wiley.
- Bramer, M. (2016). *Principles of Data Mining*. Portsmouth: Springer.
- Brous, P. J. (2020). The dual effects of the Internet of Things (IoT): A systematic review of the benefits and risks of IoT adoption by organizations. *International Journal of Information Management*, 101952.

- Cadden, A., & Keelor, B. (2017). *Implementation and Transition of Data Interchange for Geotechnical and Geoenvironmental Specialists*. ASCE-GI.
- Chatfield, C. (1996). *The Analysis of Time Series: An Introduction*. Chapman and Hall.
- Christian, J. B. (2011). Unresolved Problems in Geotechnical Risk and Reliability. *Geotechnical Special Publication*, 50-63.
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, Vol. 20, Issue 3*, 273-297.
- Cryer, J. C. (2008). *Time series analysis*. Iowa City: Springer.
- Cui, X. e. (2018). Laboratory tests on the engineering properties of sensor-enabled geobelts. *Geotextiles and Geomembranes*, 46, 66-76.
- EDF. (1989-2020). Finite Element Analysis of Structures and Thermomechanics for Studies and Research. www.code-aster.org.
- EU BIM Taskgroup. (2017). *Handbook for the Introduction of Building Information Modeling by the European Public Sector*. EU BIM Taskgroup.
- Fabozzi S., B. S. (2020). I-BIM based approach for geotechnical and numerical modelling of a conventional tunnel excavation. *Tunnelling and Underground Space Technology*, 108.
- Fardipour, M. e. (2016). Interaction analysis of waffle slabs supporting houses on expansive soil. *Innov. Infrastruct. Solut.*, 1(20).
- Floridi, L. T. (2016, December). What is data ethics? *Philosophical Transactions of The Royal Society A Mathematical Physical and Engineering Sciences*, 374.

- Goldberger, J., Hinton, G., Roweis, S., & Salakhutdinov, R. (2005). Neighbourhood Components Analysis. *Advances in Neural Information Processing Systems, Vol 17*, 513-520.
- Goodman, B. F. (2017). European Union regulations on algorithmic decision-making. *AI Magazine, Vol 38, No 3*.
- Halko, N., Martinsson, P., & Tropp, J. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *IAM Rev., Survey and Review section, Vol. 53, num. 2*, 217-288.
- Huang M. Q., N. J. (2020). BIM, machine learning and computer vision techniques in underground. *Tunneling and Underground Space Technology Incorporating Trenchless Technology Research*, 103677.
- Hyperledger. (2017). *Hyperledger Architecture, Volume 1*. Retrieved from Hyperledger website: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf
- J. Rio, B. F.-M. (2013, Apr/Jun). Expansion of IFC model with structural sensors. *Informe de la Construccion*, pp. 219-228.
- Jones, E., Oliphant, E., & Peterson, P. (2001). SciPy: Open Source Scientific Tools for Python. Retrieved from <http://www.scipy.org/>
- Karunaratne, A. e. (2012). Review of residential footing design on expansive soil in Australia. In A. M. Samali B., *Form materials to structures: Advancement through innovation* (pp. 575-580). CRC Press.
- Kazymyrenko, K., & Laverne, J. (2009). Advancement de la modélisation mécanique des joints-plots. *CR-AMA.09.039*.

- Koerner, R. e. (2019). *Relative Sustainability (i.e., Embodied Carbon) Calculations With Respect*. Folsom: Geosynthetic Institute.
- Kubat, M. (2017). *An introduction to machine learning*. Coral Gables: Springer.
- Lajnef, N. e. (2013). *Smart Pavement Monitoring System*. Washington, DC: Federal Highway Administration - Office of Acquisition Management.
- Leshchinsky, B., & Ling, H. (2012). Numerical modeling of behavior of railway ballasted structure with geocell confinement. *Geotextiles and Geomembranes*.
- Li W., B. M. (2020). Real-time GIS for smart cities. *International Journal of Geographical Information Science*, 311-324.
- Liu, X., Wang, X., Wright, G., Cheng, J., Li, X., & Liu, R. (2017). A State-of-the-Art Review on the Integration of Building Information Modeling (BIM) and Geographic Information System (GIS). *ISPRS Int. J. Geo-Inf.*, 6(53).
- Liu, Z., Gilbert, G., Cepada, J., Lysdahl, A., Paciullo, L., Hefre, H., & Lacasse, S. (2020). Modeling of shallow landslides with Machine Learning Algorithms. *Geoscience Frontiers*, 12.
- Lutkepohl, H. (1993). *Introduction to Multiple Time Series Analysis*. Springer Verlag.
- Madou, M. J. (2011). Vol II, Manufacturing Techniques for Microfabrication and Nanotechnology. In M. J. Madou, *Fundamentals of Microfabrication and Nanotechnology* (pp. 160-167). CRC Press Taylor & Francis Group.
- Masino J, T. J. (2017). Learning from the crowd: Road infrastructure monitoring system. *Traffic and Transportation Engineering*, 451-463.
- Microsoft. (2019). *Microsoft Cognitive Toolkit*. Retrieved from <https://docs.microsoft.com/en-us/cognitive-toolkit/>

- Millman, K. J., & Aivazis, M. (2011). Python for Scientists and Engineers. *Computing in Science & Engineering*, 13, 9-12.
- Minde, R. R. (2018). Analyzing the factors influencing quality throughout the lifecycle of a road project. *3rd International conference on Construction, Real estate, Infrastructure and Project management*. Pune.
- Muratov F., L. A. (2018). YAC: BFT Consensus Algorithm for Blockchain. Sora Decentralized Autonomous Economy Project.
- Nizar L., K. C. (2013). *Smart Pavement Monitoring System*. Washington, DC: Federal Highway Administration.
- Oliphant, T. E. (2007). Python for Scientific Computing. *Computing in Science & Engineering*, 9, 10-20.
- O'Regan, G. (2017). *Concise Guide to Software Engineering*. Springer.
- Paik H., X. X. (2019). Analysis of Data Management in Blockchain-Based Systems: From Architecture to Governance. *IEEE Access (vol. 7)*, 186091-186107.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Perrot, M. (2011). Scikit-learn: Machine Learning in Python. *JMLR 12*, 2825-2830. Retrieved from <https://scikit-learn.org/stable/>
- Penny, W. (2000). *Signal Processing Course*. Penny, W.D.
- Poirer, E. A.-F. (2014). Dimensions of Interoperability in the AEC Industry. *ASCE Construction Research Congress 2014* (pp. 1987-1996). ASCE.
- Pokharel, S. H. (2008). Experimental evaluation of geocell-reinforced bases under repeated loading. *Int. J. of Pavement Research and Technology, issue 11*, 114-127.

- Porter, M. E. (1998). *Competitive Advantage: Creating and Sustaining Superior Performance*. Free Press.
- Porter, M. E. (1998). *Competitive Strategy: Techniques for Analyzing Industries and Competitors*. Free Press.
- Proakis, J., & Manolakis, D. (1996). *Digital Signal Processing: Principles, Algorithms and Applications*. Prentice-Hall.
- Rayes, A. S. (2019). *Internet of Things from Hype to Reality*. San Jose: Springer.
- Riaz, Z. P. (2017). BIM and sensors-based data management system for construction safety monitoring. *J. of Eng. Design and Technology*, 15,6, 738-753.
- Rio, J., Ferreira, B., & Martins, J. (2013). Expansion of IFC model with structural sensors. *Informes de la Construcción*, 219-228.
- Schwab, K. (2019). *The Global Competitiveness Report*. Geneva: World Economic Forum.
- Seraj, F. e. (2016). RoADS: A Road Pavement Monitoring System for Anomaly Detection using Smart Phones. *Urban Sensor Data, SenseML 2014, Revised Selected Papers* (pp. 128-146). Springer.
- Seraj, F., Zwaag, B. v., Dilo, A., Luarasi, T., & Havinga, P. (2016). RoADS: A Road Pavement Monitoring System. *Conference: International Workshop on Modeling Social Media International Workshop on Mining Ubiquitous and Social Environments International Workshop on Machine Learning for Urban Sensor Data*. Springer International Publishing Switzerland.

- Shams, M. I. (2018). Analysis and Modeling of Stiffened Slab Foundation on Expansive Soils. *International Congress and Exhibition "Sustainable Civil Infrastructures: Innovative Infrastructure Geotechnology*, (pp. 250-261).
- Smarsly K., T. E. (2016). Monitoring information modeling for semantic mapping of structural health monitoring. *The 16th International Conference on Computing in Civil and Building*. Osaka, Japan.
- Stucki, M. e. (2011). *Comparative Life Cycle Assessment of Geosynthetics versus Conventional*. Zurich: European Association for Geosynthetic Manufacturers (EAGM).
- Tawelian, L. R. (2016). The Implementation of Geotechnical Data Into the BIM Process. *The 3rd International Conference on Transportation Geotechnics (ICTG 2016)* (pp. 734-741). Elsevier B.V.
- Wang, H. (2013). Integration of BIM and live sensing information to monitor building energy performance. *30th CIB W78 International Conference - October 9-12*, (pp. 344-352). Beijing.
- WHO. (2018). *Global status report on road safety*. Geneva: World Health Organization.
- Xue W., W. L. (2015). A prototype integrated monitoring system for pavement and traffic based on an embedded sensing network. *IEEE Transactions on Intelligent Transportation Systems*, 1380-1390.
- Yazdani, H., & Hatami, K. (2016). Sensor-Enabled Geogrids for Stabilization and Performance. *Int. J. of Geosynth. and Ground Eng.*, 37-43.
- Zuboff, S. (2018). *The Age of Surveillance Capitalism*. Public Affairs.

Annex 1

A1.1 Hardware description

The hardware prototype is based upon two main off-the-shelf components:

- Freescale Semiconductor, Inc. Xtrinsic MMA8451Q 3-Axis, 14-bit/8-bit Digital Accelerometer (Adafruit MMA8451 board)
- Raspberry Pi 3 Model B+, 1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernet support

The lab prototype has been tested on both power-over-ethernet and external power supply configurations. The data stream has been managed through our facility's network and via a dedicated Linux server node (dedicated to data storage and cloud-computing procedures testing).

Enclosing and wiring has been performed at the most basic level through commercially available components.

In the following pages the technical spec-sheets of both 3-Axis accelerometer and Raspberry Pi board are reproduced.

(official spec-sheets follow)

Xtrinsic MMA8451Q 3-Axis, 14-bit/8-bit Digital Accelerometer

The MMA8451Q is a smart, low-power, three-axis, capacitive, micromachined accelerometer with 14 bits of resolution. This accelerometer is packed with embedded functions with flexible user programmable options, configurable to two interrupt pins. Embedded interrupt functions allow for overall power savings relieving the host processor from continuously polling data. There is access to both low-pass filtered data as well as high-pass filtered data, which minimizes the data analysis required for jolt detection and faster transitions. The device can be configured to generate inertial wakeup interrupt signals from any combination of the configurable embedded functions allowing the MMA8451Q to monitor events and remain in a low-power mode during periods of inactivity. The MMA8451Q is available in a 3 mm by 3 mm by 1 mm QFN package.

Features

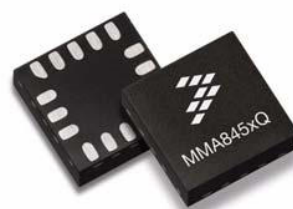
- 1.95V to 3.6V supply voltage
- 1.6V to 3.6V interface voltage
- $\pm 2g/\pm 4g/\pm 8g$ dynamically selectable full-scale
- Output Data Rates (ODR) from 1.56 Hz to 800 Hz
- 99 $\mu g/\sqrt{Hz}$ noise
- 14-bit and 8-bit digital output
- I²C digital output interface (operates to 2.25 MHz with 4.7 k Ω pullup)
- Two programmable interrupt pins for seven interrupt sources
- Three embedded channels of motion detection
- Freefall or Motion Detection: 1 channel
- Pulse Detection: 1 channel
- Jolt Detection: 1 channel
- Orientation (Portrait/Landscape) detection with programmable hysteresis
- Automatic ODR change for Auto-WAKE and return to SLEEP
- 32-sample FIFO
- High-Pass Filter Data available per sample and through the FIFO
- Self-Test
- RoHS compliant
- Current Consumption: 6 μA to 165 μA

Typical Applications

- E-Compass applications
- Static orientation detection (Portrait/Landscape, Up/Down, Left/Right, Back/Front position identification)
- Notebook, e-reader, and Laptop Tumble and Freefall Detection
- Real-time orientation detection (virtual reality and gaming 3D user position feedback)
- Real-time activity analysis (pedometer step counting, freefall drop detection for HDD, dead-reckoning GPS backup)
- Motion detection for portable product power saving (Auto-SLEEP and Auto-WAKE for cell phone, PDA, GPS, gaming)
- Shock and vibration monitoring (mechatronic compensation, shipping and warranty usage logging)
- User interface (menu scrolling by orientation change, tap detection for button replacement)

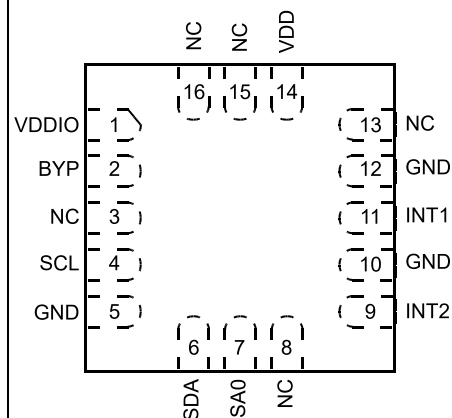
MMA8451Q

Top and Bottom View



16 PIN QFN
3 mm by 3 mm by 1 mm
CASE 2077-02

Top View



Pin Connections

ORDERING INFORMATION

Part Number	Temperature Range	Package Description	Shipping
MMA8451QT	-40°C to +85°C	QFN-16	Tray
MMA8451QR1	-40°C to +85°C	QFN-16	Tape and Reel

Contents

1	Block Diagram and Pin Description	3
1.1	Soldering Information	5
2	Mechanical and Electrical Specifications	6
2.1	Mechanical Characteristics	6
2.2	Electrical Characteristics	7
2.3	I ² C interface characteristics	8
2.4	Absolute Maximum Ratings	9
3	Terminology	10
3.1	Sensitivity	10
3.2	Zero-g Offset	10
3.3	Self-Test	10
4	Modes of Operation	11
5	Functionality	12
5.1	Device Calibration	12
5.2	8-bit or 14-bit Data	13
5.3	Internal FIFO Data Buffer	13
5.4	Low Power Modes vs. High Resolution Modes	13
5.5	Auto-WAKE/SLEEP Mode	13
5.6	Freefall and Motion Detection	14
5.7	Transient Detection	14
5.8	Tap Detection	14
5.9	Orientation Detection	15
5.10	Interrupt Register Configurations	16
5.11	Serial I ² C Interface	16
6	Register Descriptions	19
6.1	Data Registers	20
6.2	32 Sample FIFO	22
6.3	Portrait/Landscape Embedded Function Registers	27
6.4	Motion and Freefall Embedded Function Registers	30
6.5	Transient (HPF) Acceleration Detection	35
6.6	Single, Double and Directional Tap Detection Registers	37
6.7	Auto-WAKE/SLEEP Detection	41
6.8	Control Registers	42
6.9	User Offset Correction Registers	45

Related Documentation

The MMA8451Q device features and operations are described in a variety of reference manuals, user guides, and application notes. To find the most-current versions of these documents:

1. Go to the Freescale homepage at:
<http://www.freescale.com/>
2. In the Keyword search box at the top of the page, enter the device number MMA8451Q.
3. In the Refine Your Result pane on the left, click on the Documentation link.

MMA8451Q

1 Block Diagram and Pin Description

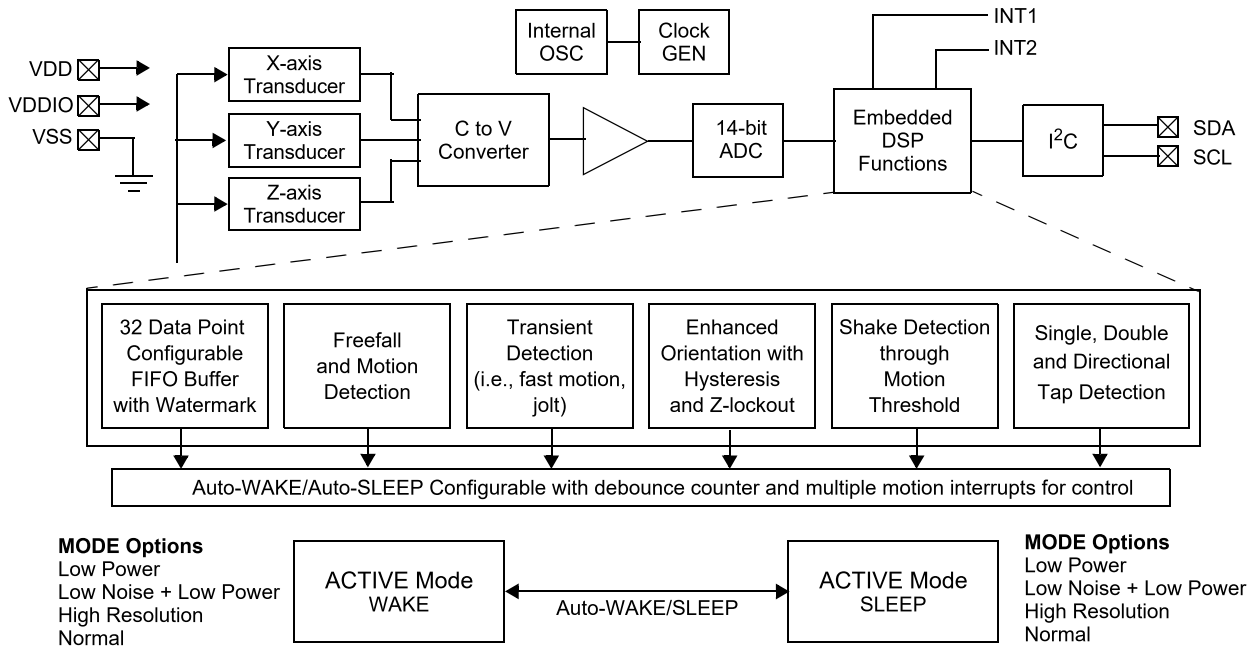


Figure 1. Block Diagram

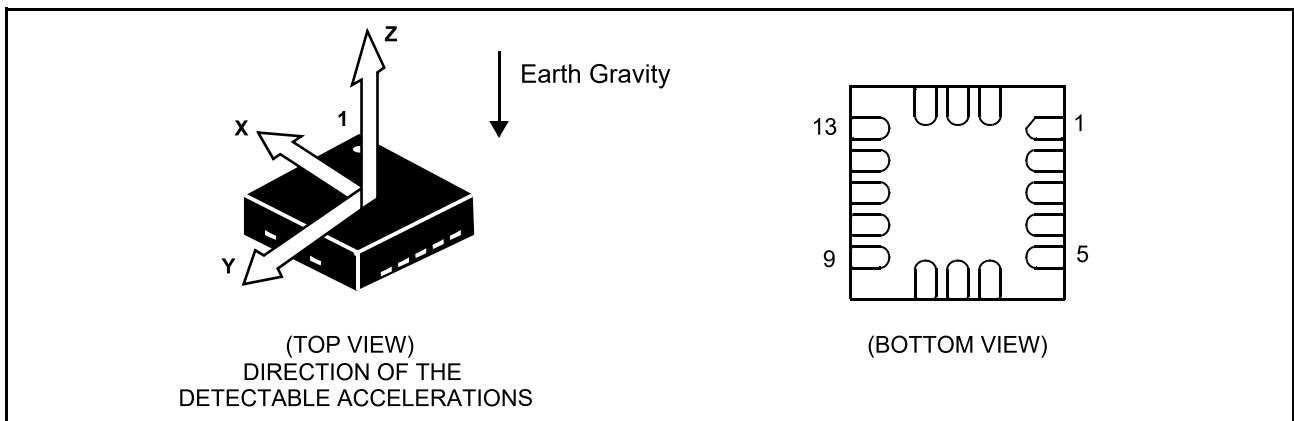


Figure 2. Direction of the Detectable Accelerations

Figure 3 shows the device configuration in the 6 different orientation modes. These orientations are defined as the following: PU = Portrait Up, LR = Landscape Right, PD = Portrait Down, LL = Landscape Left, BACK and FRONT side views. There are several registers to configure the orientation detection and are described in detail in the register setting section.

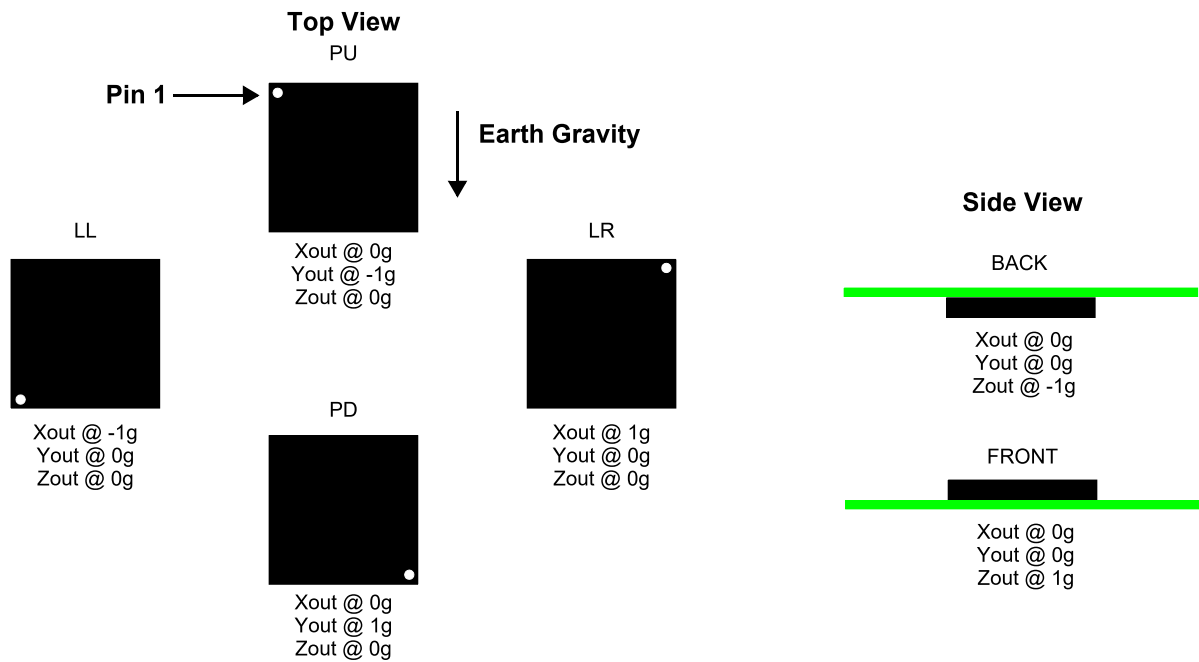


Figure 3. Landscape/Portrait Orientation

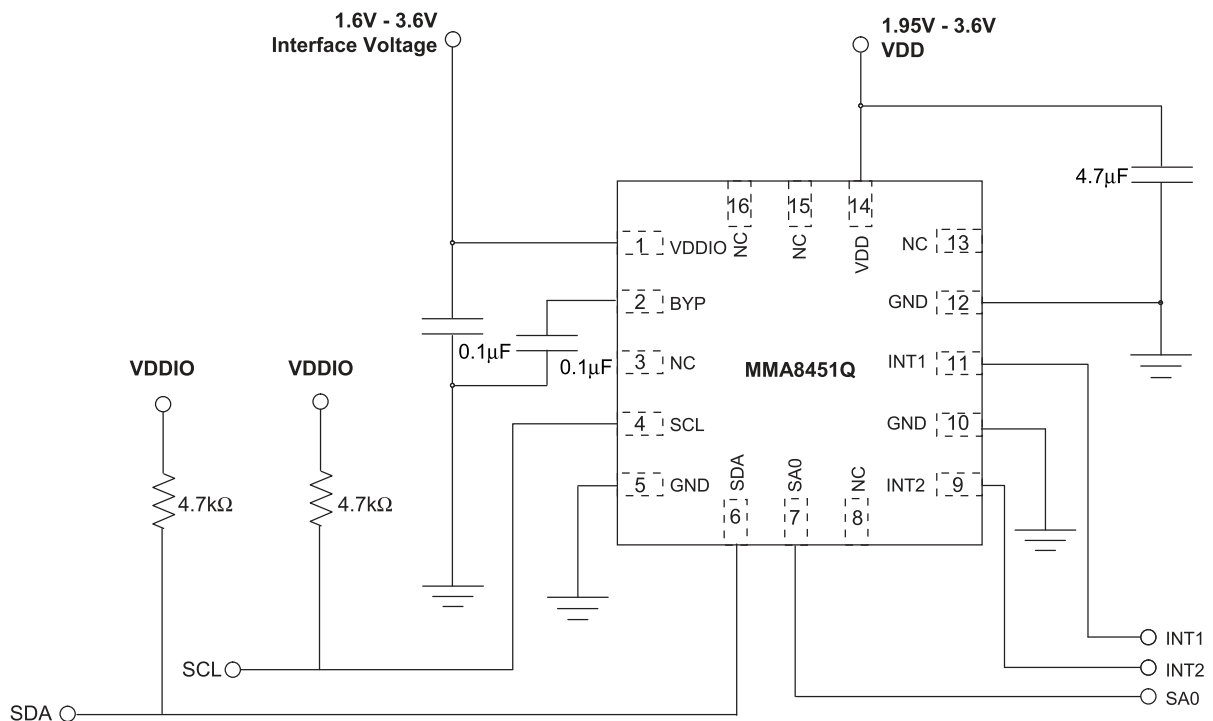


Figure 4. Application Diagram

Table 1. Pin Description

Pin #	Pin Name	Description	Pin Status
1	VDDIO	Internal Power Supply (1.62V - 3.6V)	Input
2	BYP	Bypass capacitor (0.1 μ F)	Input
3	NC	Leave open. Do not connect	Open
4	SCL	I ² C Serial Clock	Open Drain
5	GND	Connect to Ground	Input
6	SDA	I ² C Serial Data	Open Drain
7	SA0	I ² C Least Significant Bit of the Device I ² C Address	Input
8	NC	Internally not connected (can be GND or VDD)	Input
9	INT2	Inertial Interrupt 2	Output
10	GND	Connect to Ground	Input
11	INT1	Inertial Interrupt 1	Output
12	GND	Connect to Ground	Input
13	NC	Internally not connected (can be GND or VDD)	Input
14	VDD	Power Supply (1.95V to 3.6V)	Input
15	NC	Internally not connected (can be GND or VDD)	Input
16	NC	Internally not connected (can be GND or VDD)	Input

The device power is supplied through VDD line. Power supply decoupling capacitors (100 nF ceramic plus 4.7 μ F bulk, or a single 4.7 μ F ceramic) should be placed as near as possible to the pins 1 and 14 of the device.

The control signals SCL, SDA, and SA0 are not tolerant of voltages more than VDDIO + 0.3V. If VDDIO is removed, the control signals SCL, SDA, and SA0 will clamp any logic signals with their internal ESD protection diodes.

The functions, the threshold and the timing of the two interrupt pins (INT1 and INT2) are user programmable through the I²C interface. The SDA and SCL I²C connections are open drain and therefore require a pullup resistor as shown in the application diagram in [Figure 4](#).

1.1 Soldering Information

The QFN package is compliant with the RoHS standard. Please refer to AN4077.

2 Mechanical and Electrical Specifications

2.1 Mechanical Characteristics

Table 2. Mechanical Characteristics @ VDD = 2.5V, VDDIO = 1.8V, T = 25°C unless otherwise noted.

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Measurement Range ⁽¹⁾	FS[1:0] set to 00 2g Mode	FS		±2		g
	FS[1:0] set to 01 4g Mode			±4		
	FS[1:0] set to 10 8g Mode			±8		
Sensitivity	FS[1:0] set to 00 2g Mode	So		4096		counts/g
	FS[1:0] set to 01 4g Mode			2048		
	FS[1:0] set to 10 8g Mode			1024		
Sensitivity Accuracy ⁽²⁾		Soa		±2.64		%
Sensitivity Change vs. Temperature	FS[1:0] set to 00 2g Mode	TCS _o		±0.008		%/ ^o C
	FS[1:0] set to 01 4g Mode					
	FS[1:0] set to 10 8g Mode					
Zero-g Level Offset Accuracy ⁽³⁾	FS[1:0] 2g, 4g, 8g	TyOff		±17		mg
Zero-g Level Offset Accuracy Post Board Mount ⁽⁴⁾	FS[1:0] 2g, 4g, 8g	TyOffPBM		±20		mg
Zero-g Level Change vs. Temperature	-40°C to 85°C	TCOff		±0.15		mg/ ^o C
Self-Test Output Change ⁽⁵⁾ X Y Z	FS[1:0] set to 0 4g Mode	Vst		+181 +255 +1680		LSB
ODR Accuracy 2 MHz Clock				±2		%
Output Data Bandwidth		BW	ODR/3		ODR/2	Hz
Output Noise	Normal Mode ODR = 400 Hz	Noise		126		µg/ [√] Hz
Output Noise Low Noise Mode ⁽¹⁾	Normal Mode ODR = 400 Hz	Noise		99		µg/ [√] Hz
Operating Temperature Range		Top	-40		+85	°C

- Dynamic Range is limited to 4g when the Low Noise bit in Register 0x2A, bit 2 is set.
- Sensitivity remains in spec as stated, but changing Oversampling mode to Low Power causes 3% sensitivity shift. This behavior is also seen when changing from 800 Hz to any other data rate in the Normal, Low Noise + Low Power or High Resolution mode.
- Before board mount.
- Post Board Mount Offset Specifications are based on an 8 Layer PCB, relative to 25°C.
- Self-Test is one direction only.

2.2 Electrical Characteristics

Table 3. Electrical Characteristics @ VDD = 2.5V, VDDIO = 1.8V, T = 25°C unless otherwise noted.

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Supply Voltage		VDD ⁽¹⁾	1.95	2.5	3.6	V
Interface Supply Voltage		VDDIO ⁽¹⁾	1.62	1.8	3.6	V
Low Power Mode	ODR = 1.56 Hz	I _{ddLP}		6		μA
	ODR = 6.25 Hz			6		
	ODR = 12.5 Hz			6		
	ODR = 50 Hz			14		
	ODR = 100 Hz			24		
	ODR = 200 Hz			44		
	ODR = 400 Hz			85		
Normal Mode	ODR = 800 Hz	I _{dd}		165		μA
	ODR = 1.56 Hz			24		
	ODR = 6.25 Hz			24		
	ODR = 12.5 Hz			24		
	ODR = 50 Hz			24		
	ODR = 100 Hz			44		
	ODR = 200 Hz			85		
Current during Boot Sequence, 0.5 mSec max duration using recommended Bypass Cap	VDD = 2.5V	I _{dd Boot}			1	mA
Value of Capacitor on BYP Pin	-40°C 85°C	Cap	75	100	470	nF
STANDBY Mode Current @25°C	VDD = 2.5V, VDDIO = 1.8V STANDBY Mode	I _{ddStby}		1.8	5	μA
Digital High Level Input Voltage SCL, SDA, SA0		V _{IH}	0.75*VDDIO			V
Digital Low Level Input Voltage SCL, SDA, SA0		V _{IL}			0.3*VDDIO	V
High Level Output Voltage INT1, INT2	I _O = 500 μA	V _{OH}	0.9*VDDIO			V
Low Level Output Voltage INT1, INT2	I _O = 500 μA	V _{OL}			0.1*VDDIO	V
Low Level Output Voltage SDA	I _O = 500 μA	V _{OLS}			0.1*VDDIO	V
Power on Ramp Time			0.001		1000	ms
Boot time	Time from VDDIO on and VDD > VDD min until I ² C is ready for operation, C _{byp} = 100 nF	T _{bt}		350	500	μs
Turn-on time	Time to obtain valid data from STANDBY mode to ACTIVE mode.	T _{on1}			2/ODR + 1 ms	
Turn-on time	Time to obtain valid data from valid voltage applied.	T _{on2}			2/ODR + 2 ms	
Operating Temperature Range		T _{op}	-40		+85	°C

1. There is no requirement for power supply sequencing. The VDDIO input voltage can be higher than the VDD input voltage.

2.3 I²C interface characteristics

Table 4. I²C slave timing values⁽¹⁾

Parameter	Symbol	I ² C Fast Mode		Unit
		Min	Max	
SCL clock frequency	f_{SCL}	0	400	kHz
Bus-free time between STOP and START condition	t_{BUF}	1.3		μ s
(Repeated) START hold time	$t_{HD;STA}$	0.6		μ s
Repeated START setup time	$t_{SU;STA}$	0.6		μ s
STOP condition setup time	$t_{SU;STO}$	0.6		μ s
SDA data hold time	$t_{HD;DAT}$	0.05	0.9 ⁽²⁾	μ s
SDA setup time	$t_{SU;DAT}$	100		ns
SCL clock low time	t_{LOW}	1.3		μ s
SCL clock high time	t_{HIGH}	0.6		μ s
SDA and SCL rise time	t_r	$20 + 0.1 C_b^{(3)}$	300	ns
SDA and SCL fall time	t_f	$20 + 0.1 C_b^{(3)}$	300	ns
SDA valid time ⁽⁴⁾	$t_{VD;DAT}$		0.9 ⁽²⁾	μ s
SDA valid acknowledge time ⁽⁵⁾	$t_{VD;ACK}$		0.9 ⁽²⁾	μ s
Pulse width of spikes on SDA and SCL that must be suppressed by internal input filter	t_{SP}	0	50	ns
Capacitive load for each bus line	C_b		400	pF

1. All values referred to $V_{IH(min)}$ ($0.3V_{DD}$) and $V_{IL(max)}$ ($0.7V_{DD}$) levels.

2. This device does not stretch the LOW period (t_{LOW}) of the SCL signal.

3. C_b = total capacitance of one bus line in pF.

4. $t_{VD;DAT}$ = time for data signal from SCL LOW to SDA output (HIGH or LOW, depending on which one is worse).

5. $t_{VD;ACK}$ = time for Acknowledgement signal from SCL LOW to SDA output (HIGH or LOW, depending on which one is worse).

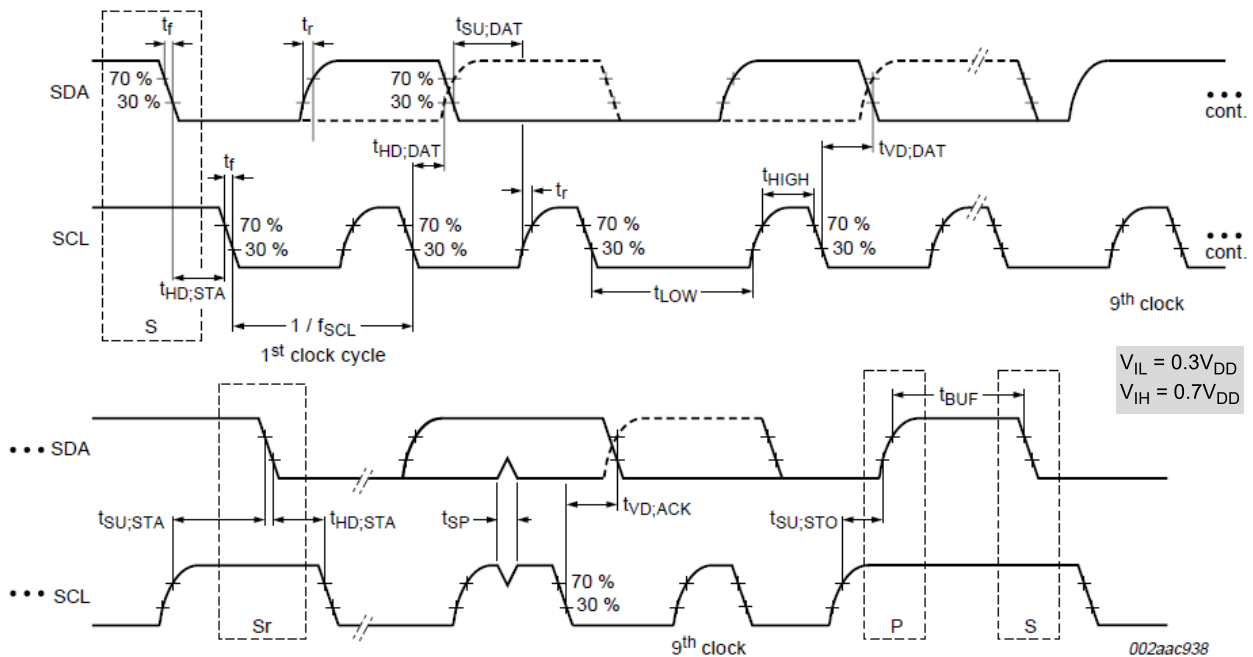


Figure 5. I²C slave timing diagram

2.4 Absolute Maximum Ratings

Stresses above those listed as “absolute maximum ratings” may cause permanent damage to the device. Exposure to maximum rating conditions for extended periods may affect device reliability.

Table 5. Maximum Ratings

Rating	Symbol	Value	Unit
Maximum Acceleration (all axes, 100 μ s)	g_{max}	5,000	g
Supply Voltage	VDD	-0.3 to + 3.6	V
Input voltage on any control pin (SA0, SCL, SDA)	Vin	-0.3 to VDDIO + 0.3	V
Drop Test	D_{drop}	1.8	m
Operating Temperature Range	T_{OP}	-40 to +85	$^{\circ}$ C
Storage Temperature Range	T_{STG}	-40 to +125	$^{\circ}$ C

Table 6. ESD and Latchup Protection Characteristics

Rating	Symbol	Value	Unit
Human Body Model	HBM	\pm 2000	V
Machine Model	MM	\pm 200	V
Charge Device Model	CDM	\pm 500	V
Latchup Current at T = 85 $^{\circ}$ C	—	\pm 100	mA



This device is sensitive to mechanical shock. Improper handling can cause permanent damage of the part or cause the part to otherwise fail.



This device is sensitive to ESD, improper handling can cause permanent damage to the part.

3 Terminology

3.1 Sensitivity

The sensitivity is represented in counts/g. In 2g mode the sensitivity is 4096 counts/g. In 4g mode the sensitivity is 2048 counts/g and in 8g mode the sensitivity is 1024 counts/g.

3.2 Zero-g Offset

Zero-g Offset (TyOff) describes the deviation of an actual output signal from the ideal output signal if the sensor is stationary. A sensor stationary on a horizontal surface will measure 0g in X-axis and 0g in Y-axis whereas the Z-axis will measure 1g. The output is ideally in the middle of the dynamic range of the sensor (content of OUT Registers 0x00, data expressed as 2's complement number). A deviation from ideal value in this case is called Zero-g offset. Offset is to some extent a result of stress on the MEMS sensor and therefore the offset can slightly change after mounting the sensor onto a printed circuit board or exposing it to extensive mechanical stress.

3.3 Self-Test

Self-Test checks the transducer functionality without external mechanical stimulus. When Self-Test is activated, an electrostatic actuation force is applied to the sensor, simulating a small acceleration. In this case, the sensor outputs will exhibit a change in their DC levels which are related to the selected full scale through the device sensitivity. When Self-Test is activated, the device output level is given by the algebraic sum of the signals produced by the acceleration acting on the sensor and by the electrostatic test-force.

4 Modes of Operation

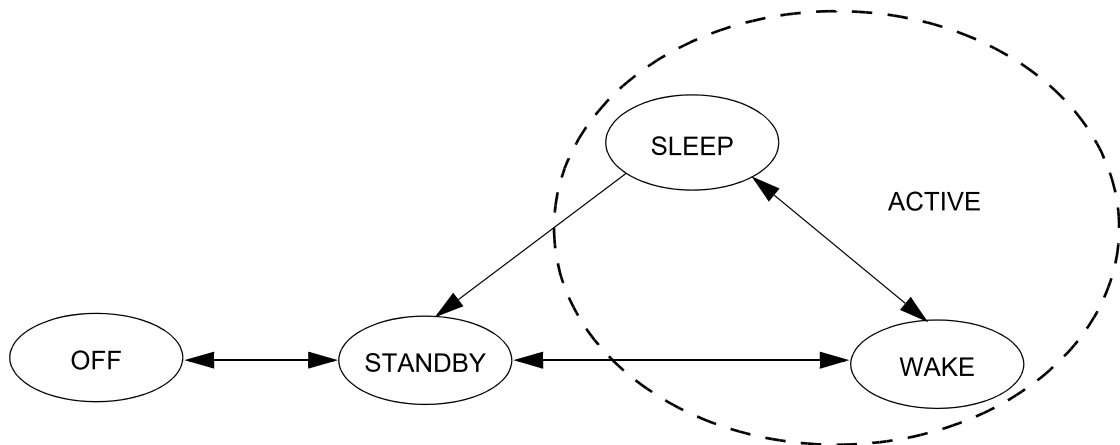


Figure 6. MMA8451Q Mode Transition Diagram

Table 7. Mode of Operation Description

Mode	I ² C Bus State	VDD	VDDIO	Function Description
OFF	Powered Down	<1.8V	VDDIO Can be > VDD	The device is powered off. All analog and digital blocks are shutdown. I ² C bus inhibited.
STANDBY	I ² C communication with MMA8451Q is possible	ON	VDDIO = High VDD = High ACTIVE bit is cleared	Only digital blocks are enabled. Analog subsystem is disabled. Internal clocks disabled.
ACTIVE (WAKE/SLEEP)	I ² C communication with MMA8451Q is possible	ON	VDDIO = High VDD = High ACTIVE bit is set	All blocks are enabled (digital, analog).

All register contents are preserved when transitioning from ACTIVE to STANDBY mode. Some registers are reset when transitioning from STANDBY to ACTIVE. These are all noted in the device memory map register table. The SLEEP and WAKE modes are ACTIVE modes. For more information on how to use the SLEEP and WAKE modes and how to transition between these modes, please refer to the functionality section of this document.

5 Functionality

The MMA8451Q is a low-power, digital output 3-axis linear accelerometer with a I²C interface and embedded logic used to detect events and notify an external microprocessor over interrupt lines. The functionality includes the following:

- 8-bit or 14-bit data, High-Pass Filtered data, 8-bit or 14-bit configurable 32 sample FIFO
- Four different oversampling options for compromising between resolution and current consumption based on application requirements
- Additional Low Noise mode that functions independently of the Oversampling modes for higher resolution
- Low Power and Auto-WAKE/SLEEP modes for conservation of current consumption
- Single/Double tap with directional information 1 channel
- Motion detection with directional information or Freefall 1 channel
- Transient/Jolt detection based on a high-pass filter and settable threshold for detecting the change in acceleration above a threshold with directional information 1 channel
- Flexible user configurable portrait landscape detection algorithm addressing many use cases for screen orientation

All functionality is available in 2g, 4g or 8g dynamic ranges. There are many configuration settings for enabling all the different functions. Separate application notes have been provided to help configure the device for each embedded functionality.

Table 8. Features of the MMA845xQ devices

Feature List	MMA8451	MMA8452	MMA8453
Digital Resolution (Bits)	14	12	10
Digital Sensitivity (Counts/g)	4096	1024	256
Data-Ready Interrupt	Yes	Yes	Yes
Single-Pulse Interrupt	Yes	Yes	Yes
Double-Pulse Interrupt	Yes	Yes	Yes
Directional-Pulse Interrupt	Yes	Yes	Yes
Auto-WAKE	Yes	Yes	Yes
Auto-SLEEP	Yes	Yes	Yes
Freefall Interrupt	Yes	Yes	Yes
32 Level FIFO	Yes	No	No
High-Pass Filter	Yes	Yes	Yes
Low-Pass Filter	Yes	Yes	Yes
Orientation Detection Portrait/Landscape = 30°, Landscape to Portrait = 60°, and Fixed 45° Threshold	Yes	Yes	Yes
Programmable Orientation Detection	Yes	No	No
Motion Interrupt with Direction	Yes	Yes	Yes
Transient Detection with High-Pass Filter	Yes	Yes	Yes
Low Power Mode	Yes	Yes	Yes

5.1 Device Calibration

The device interface is factory calibrated for sensitivity and Zero-g offset for each axis. The trim values are stored in Non Volatile Memory (NVM). On power-up, the trim parameters are read from NVM and applied to the circuitry. In normal use, further calibration in the end application is not necessary. However, the MMA8451Q allows the user to adjust the Zero-g offset for each axis after power-up, changing the default offset values. The user offset adjustments are stored in 6 volatile registers. For more information on device calibration, refer to Freescale application note, AN4069.

5.2 8-bit or 14-bit Data

The measured acceleration data is stored in the OUT_X_MSB, OUT_X_LSB, OUT_Y_MSB, OUT_Y_LSB, OUT_Z_MSB, and OUT_Z_LSB registers as 2's complement 14-bit numbers. The most significant 8-bits of each axis are stored in OUT_X (Y, Z)_MSB, so applications needing only 8-bit results can use these 3 registers and ignore OUT_X,Y,Z_LSB. To do this, the F_READ bit in CTRL_REG1 must be set. When the F_READ bit is cleared, the fast read mode is disabled.

When the full-scale is set to 2g, the measurement range is -2g to +1.99975g, and each count corresponds to 1g/4096 (0.25 mg) at 14-bits resolution. When the full-scale is set to 8g, the measurement range is -8g to +7.999g, and each count corresponds to 1g/1024 (0.98 mg) at 14-bits resolution. The resolution is reduced by a factor of 64 if only the 8-bit results are used. For more information on the data manipulation between data formats and modes, refer to Freescale application.

5.3 Internal FIFO Data Buffer

MMA8451Q contains a 32 sample internal FIFO data buffer minimizing traffic across the I²C bus. The FIFO can also provide power savings of the system by allowing the host processor/MCU to go into a SLEEP mode while the accelerometer independently stores the data, up to 32 samples per axis. The FIFO can run at all output data rates. There is the option of accessing the full 14-bit data or for accessing only the 8-bit data. When access speed is more important than high resolution the 8-bit data read is a better option.

The FIFO contains four modes (Fill Buffer Mode, Circular Buffer Mode, Trigger Mode, and Disabled Mode) described in the F_SETUP Register 0x09. Fill Buffer Mode collects the first 32 samples and asserts the overflow flag when the buffer is full and another sample arrives. It does not collect any more data until the buffer is read. This benefits data logging applications where all samples must be collected. The Circular Buffer Mode allows the buffer to be filled and then new data replaces the oldest sample in the buffer. The most recent 32 samples will be stored in the buffer. This benefits situations where the processor is waiting for an specific interrupt to signal that the data must be flushed to analyze the event. The trigger mode will hold the last data up to the point when the trigger occurs and can be set to keep a selectable number of samples after the event occurs.

The MMA8451Q FIFO Buffer has a configurable watermark, allowing the processor to be triggered after a configurable number of samples has filled in the buffer (1 to 32).

For details on the configurations for the FIFO buffer as well as more specific examples and application benefits, refer to Freescale application note, AN4073.

5.4 Low Power Modes vs. High Resolution Modes

The MMA8451Q can be optimized for lower power modes or for higher resolution of the output data. High resolution is achieved by setting the LNOISE bit in Register 0x2A. This improves the resolution but be aware that the dynamic range is limited to 4g when this bit is set. This will affect all internal functions and reduce noise. Another method for improving the resolution of the data is by oversampling. One of the oversampling schemes of the data can activated when MODS = 10 in Register 0x2B which will improve the resolution of the output data only. The highest resolution is achieved at 1.56 Hz.

There is a trade-off between low power and high resolution. Low Power can be achieved when the oversampling rate is reduced. The lowest power is achieved when MODS = 11 or when the sample rate is set to 1.56 Hz. For more information on how to configure the MMA8451Q in Low Power mode or High Resolution mode and to realize the benefits, refer to Freescale application note, AN4075.

5.5 Auto-WAKE/SLEEP Mode

The MMA8451Q can be configured to transition between sample rates (with their respective current consumption) based on four of the interrupt functions of the device. The advantage of using the Auto-WAKE/SLEEP is that the system can automatically transition to a higher sample rate (higher current consumption) when needed but spends the majority of the time in the SLEEP mode (lower current) when the device does not require higher sampling rates. Auto-WAKE refers to the device being triggered by one of the interrupt functions to transition to a higher sample rate. This may also interrupt the processor to transition from a SLEEP mode to a higher power mode.

SLEEP mode occurs after the accelerometer has not detected an interrupt for longer than the user definable time-out period. The device will transition to the specified lower sample rate. It may also alert the processor to go into a lower power mode to save on current during this period of inactivity.

The Interrupts that can WAKE the device from SLEEP are the following: Tap Detection, Orientation Detection, Motion/Freefall, and Transient Detection. The FIFO can be configured to hold the data in the buffer until it is flushed if the FIFO Gate bit is set in Register 0x2C but the FIFO cannot WAKE the device from SLEEP.

The interrupts that can keep the device from falling asleep are the same interrupts that can wake the device with the addition of the FIFO. If the FIFO interrupt is enabled and data is being accessed continually servicing the interrupt then the device will remain in the WAKE mode. Refer to AN4074, for more detailed information for configuring the Auto-WAKE/SLEEP.

5.6 Freefall and Motion Detection

MMA8451Q has flexible interrupt architecture for detecting either a Freefall or a Motion. Freefall can be enabled where the set threshold must be less than the configured threshold, or motion can be enabled where the set threshold must be greater than the threshold. The motion configuration has the option of enabling or disabling a high-pass filter to eliminate tilt data (static offset). The freefall does not use the high-pass filter. For details on the Freefall and Motion detection with specific application examples and recommended configuration settings, refer to Freescale application note, AN4070.

5.6.1 Freefall Detection

The detection of “Freefall” involves the monitoring of the X, Y, and Z axes for the condition where the acceleration magnitude is **below** a user specified threshold for a user definable amount of time. Normally, the usable threshold ranges are between ± 100 mg and ± 500 mg.

5.6.2 Motion Detection

Motion is often used to simply alert the main processor that the device is currently in use. When the acceleration exceeds a set threshold the motion interrupt is asserted. A motion can be a fast moving shake or a slow moving tilt. This will depend on the threshold and timing values configured for the event. The motion detection function can analyze static acceleration changes or faster jolts. For example, to detect that an object is spinning, all three axes would be enabled with a threshold detection of $> 2g$. This condition would need to occur for a minimum of 100 ms to ensure that the event wasn't just noise. The timing value is set by a configurable debounce counter. The debounce counter acts like a filter to determine whether the condition exists for configurable set of time (i.e., 100 ms or longer). There is also directional data available in the source register to detect the direction of the motion. This is useful for applications such as directional shake or flick, which assists with the algorithm for various gesture detections.

5.7 Transient Detection

The MMA8451Q has a built-in high-pass filter. Acceleration data goes through the high-pass filter, eliminating the offset (DC) and low frequencies. The high-pass filter cutoff frequency can be set by the user to four different frequencies which are dependent on the Output Data Rate (ODR). A higher cutoff frequency ensures the DC data or slower moving data will be filtered out, allowing only the higher frequencies to pass. The embedded Transient Detection function uses the high-pass filtered data allowing the user to set the threshold and debounce counter. The transient detection feature can be used in the same manner as the motion detection by bypassing the high-pass filter. There is an option in the configuration register to do this. This adds more flexibility to cover various customer use cases.

Many applications use the accelerometer's static acceleration readings (i.e., tilt) which measure the change in acceleration due to gravity only. These functions benefit from acceleration data being filtered with a low-pass filter where high frequency data is considered noise. However, there are many functions where the accelerometer must analyze dynamic acceleration. Functions such as tap, flick, shake and step counting are based on the analysis of the change in the acceleration. It is simpler to interpret these functions dependent on dynamic acceleration data when the static component has been removed. The Transient Detection function can be routed to either interrupt pin through bit 5 in CTRL_REG5 register (0x2E). Registers 0x1D – 0x20 are the dedicated Transient Detection configuration registers. The source register contains directional data to determine the direction of the acceleration, either positive or negative. For details on the benefits of the embedded Transient Detection function along with specific application examples and recommended configuration settings, please refer to Freescale application note, AN4071.

5.8 Tap Detection

The MMA8451Q has embedded single/double and directional tap detection. This function has various customizing timers for setting the pulse time width and the latency time between pulses. There are programmable thresholds for all three axes. The tap detection can be configured to run through the high-pass filter and also through a low-pass filter, which provides more customizing and tunable tap detection schemes. The status register provides updates on the axes where the event was detected and the direction of the tap. For more information on how to configure the device for tap detection please refer to Freescale application note, AN4072.

5.9 Orientation Detection

The MMA8451Q incorporates an advanced algorithm for orientation detection (ability to detect all 6 orientations) with configurable trip points. The embedded algorithm allows the selection of the mid point with the desired hysteresis value.

The MMA8451Q Orientation Detection algorithm confirms the reliability of the function with a configurable Z-lockout angle. Based on known functionality of linear accelerometers, it is not possible to rotate the device about the Z-axis to detect change in acceleration at slow angular speeds. The angle at which the device no longer detects the orientation change is referred to as the "Z-Lockout angle". The device operates down to 14° from the flat position.

For further information on the configuration settings of the orientation detection function, including recommendations for configuring the device to support various application use cases, refer to Freescale application note, AN4068.

Figure 8 shows the definitions of the trip angles going from Landscape to Portrait (A) and then also from Portrait to Landscape (B).

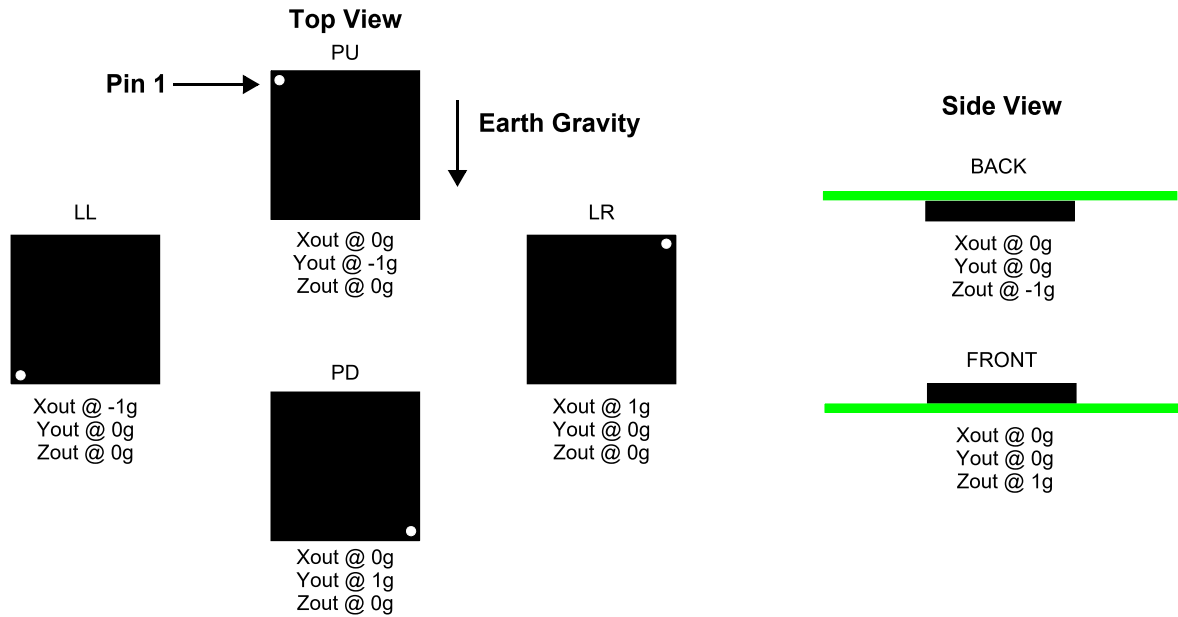


Figure 7. Landscape/Portrait Orientation

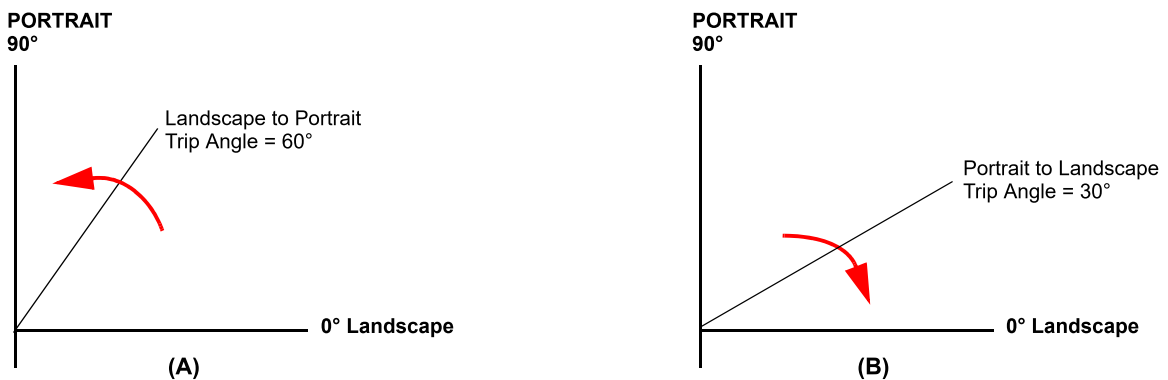


Figure 8. Illustration of Landscape to Portrait Transition (A) and Portrait to Landscape Transition (B)

Figure 9 illustrates the Z-angle lockout region. When lifting the device upright from the flat position it will be active for orientation detection as low as 14° from flat. This is user configurable. The default angle is 29° but it can be set as low as 14°.

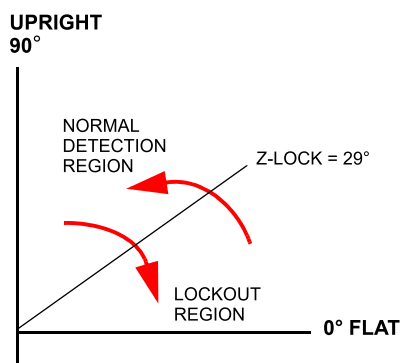


Figure 9. Illustration of Z-Tilt Angle Lockout Transition

5.10 Interrupt Register Configurations

There are seven configurable interrupts in the MMA8451Q: Data Ready, Motion/Freefall, Tap (Pulse), Orientation, Transient, FIFO and Auto-SLEEP events. These seven interrupt sources can be routed to one of two interrupt pins. The interrupt source must be enabled and configured. If the event flag is asserted because the event condition is detected, the corresponding interrupt pin, INT1 or INT2, will assert.

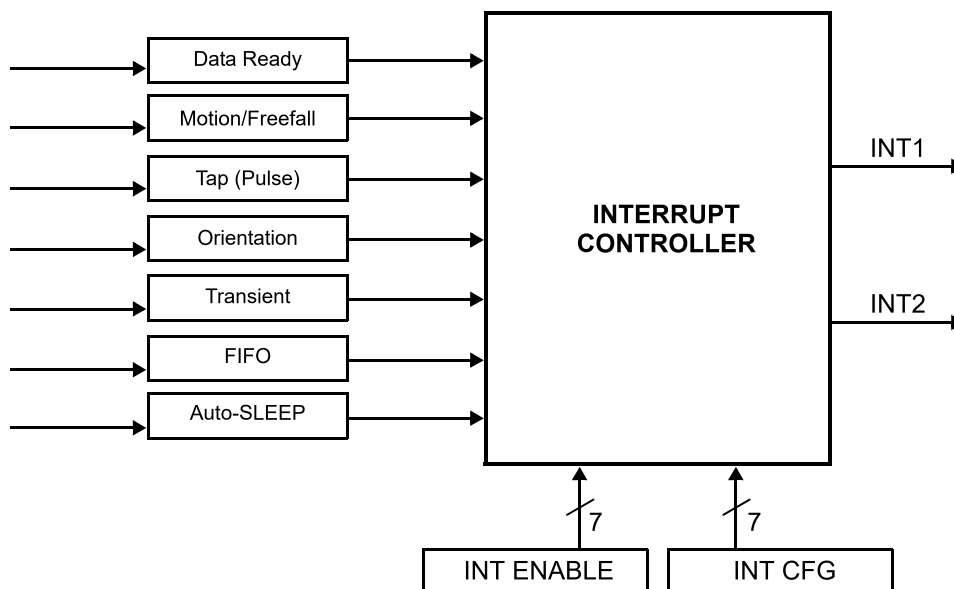


Figure 10. System Interrupt Generation Block Diagram

5.11 Serial I²C Interface

Acceleration data may be accessed through an I²C interface thus making the device particularly suitable for direct interfacing with a microcontroller. The MMA8451Q features an interrupt signal which indicates when a new set of measured acceleration data is available thus simplifying data synchronization in the digital system that uses the device. The MMA8451Q may also be configured to generate other interrupt signals accordingly to the programmable embedded functions of the device for Motion, Freefall, Transient, Orientation, and Tap.

The registers embedded inside the MMA8451Q are accessed through the I²C serial interface (Table 9). To enable the I²C interface, VDDIO line must be tied high (i.e., to the interface supply voltage). If VDD is not present and VDDIO is present, the

MMA8451Q is in off mode and communications on the I²C interface are ignored. The I²C interface may be used for communications between other I²C devices and the MMA8451Q does not affect the I²C bus.

Table 9. Serial Interface Pin Description

Pin Name	Pin Description
SCL	I ² C Serial Clock
SDA	I ² C Serial Data
SA0	I ² C least significant bit of the device address

There are two signals associated with the I²C bus; the Serial Clock Line (SCL) and the Serial Data line (SDA). The latter is a bidirectional line used for sending and receiving the data to/from the interface. External pullup resistors connected to VDDIO are expected for SDA and SCL. When the bus is free both the lines are high. The I²C interface is compliant with Fast mode (400 kHz), and Normal mode (100 kHz) I²C standards (Table 4).

5.11.1 I²C Operation

The transaction on the bus is started through a start condition (START) signal. START condition is defined as a HIGH to LOW transition on the data line while the SCL line is held HIGH. After START has been transmitted by the Master, the bus is considered busy. The next byte of data transmitted after START contains the slave address in the first 7 bits, and the eighth bit tells whether the Master is receiving data from the slave or transmitting data to the slave. When an address is sent, each device in the system compares the first seven bits after a start condition with its address. If they match, the device considers itself addressed by the Master. The 9th clock pulse, following the slave address byte (and each subsequent byte) is the acknowledge (ACK). The transmitter must release the SDA line during the ACK period. The receiver must then pull the data line low so that it remains stable low during the high period of the acknowledge clock period.

A LOW to HIGH transition on the SDA line while the SCL line is high is defined as a stop condition (STOP). A data transfer is always terminated by a STOP. A Master may also issue a repeated START during a data transfer. The MMA8451Q expects repeated STARTs to be used to randomly read from specific registers.

The MMA8451Q's standard slave address is a choice between the two sequential addresses 0011100 and 0011101. The selection is made by the high and low logic level of the SA0 (pin 7) input respectively. The slave addresses are factory programmed and alternate addresses are available at customer request. The format is shown in Table 10.

Table 10. I²C Address Selection Table

Slave Address (SA0 = 0)	Slave Address (SA0 = 1)	Comment
0011100 (0x1C)	0011101 (0x1D)	Factory Default

Single Byte Read

The MMA8451Q has an internal ADC that can sample, convert and return sensor data on request. The transmission of an 8-bit command begins on the falling edge of SCL. After the eight clock cycles are used to send the command, note that the data returned is sent with the MSB first once the data is received. Figure 11 shows the timing diagram for the accelerometer 8-bit I²C read operation. The Master (or MCU) transmits a start condition (ST) to the MMA8451Q, slave address (\$1D), with the R/W bit set to "0" for a write, and the MMA8451Q sends an acknowledgement. Then the Master (or MCU) transmits the address of the register to read and the MMA8451Q sends an acknowledgement. The Master (or MCU) transmits a repeated start condition (SR) and then addresses the MMA8451Q (\$1D) with the R/W bit set to "1" for a read from the previously selected register. The Slave then acknowledges and transmits the data from the requested register. The Master does not acknowledge (NAK) the transmitted data, but transmits a stop condition to end the data transfer.

Multiple Byte Read

When performing a multi-byte read or "burst read", the MMA8451Q automatically increments the received register address commands after a read command is received. Therefore, after following the steps of a single byte read, multiple bytes of data can be read from sequential registers after each MMA8451Q acknowledgment (AK) is received until a no acknowledge (NAK) occurs from the Master followed by a stop condition (SP) signaling an end of transmission.

Single Byte Write

To start a write command, the Master transmits a start condition (ST) to the MMA8451Q, slave address (\$1D) with the R/W bit set to "0" for a write, the MMA8451Q sends an acknowledgement. Then the Master (MCU) transmits the address of the register to write to, and the MMA8451Q sends an acknowledgement. Then the Master (or MCU) transmits the 8-bit data to write to the designated register and the MMA8451Q sends an acknowledgement that it has received the data. Since this transmission is

complete, the Master transmits a stop condition (SP) to the data transfer. The data sent to the MMA8451Q is now stored in the appropriate register.

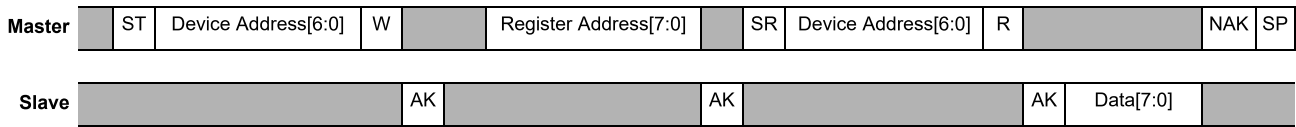
Multiple Byte Write

The MMA8451Q automatically increments the received register address commands after a write command is received. Therefore, after following the steps of a single byte write, multiple bytes of data can be written to sequential registers after each MMA8451Q acknowledgment (ACK) is received.

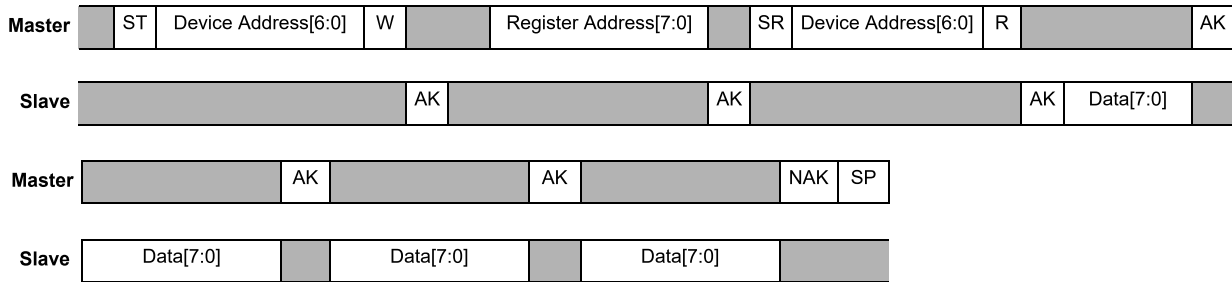
Table 11. I²C Device Address Sequence

Command	[6:1] Device Address	[0] SA0	[6:0] Device Address	R/W	8-bit Final Value
Read	001110	0	0x1C	1	0x39
Write	001110	0	0x1C	0	0x38
Read	001110	1	0x1D	1	0x3B
Write	001110	1	0x1D	0	0x3A

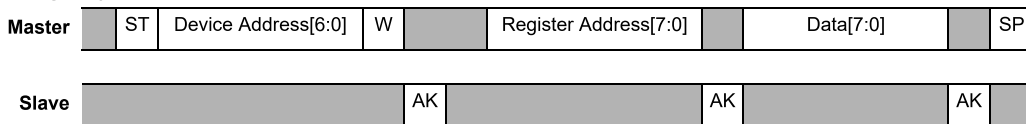
< Single Byte Read >



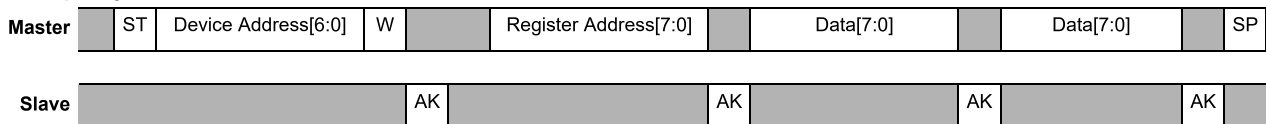
< Multiple Byte Read >



< Single Byte Write >



< Multiple Byte Write >



Legend

ST: Start Condition SP: Stop Condition NAK: No Acknowledge W: Write = 0
 SR: Repeated Start Condition AK: Acknowledge R: Read = 1

Figure 11. I²C Timing Diagram

6 Register Descriptions

Table 12. Register Address Map

Name	Type	Register Address	Auto-Increment Address				Default	Hex Value	Comment
			FMODE = 0 F_READ = 0	FMODE > 0 F_READ = 0	FMODE = 0 F_READ = 1	FMODE > 0 F_READ = 1			
STATUS/F_STATUS ⁽¹⁾⁽²⁾	R	0x00	0x01				00000000	0x00	FMODE = 0, real time status FMODE > 0, FIFO status
OUT_X_MSB ⁽¹⁾⁽²⁾	R	0x01	0x02	0x01	0x03	0x01	Output	—	[7:0] are 8 MSBs of 14-bit sample. Root pointer to XYZ FIFO data.
OUT_X_LSB ⁽¹⁾⁽²⁾	R	0x02	0x03		0x00		Output	—	[7:2] are 6 LSBs of 14-bit real-time sample
OUT_Y_MSB ⁽¹⁾⁽²⁾	R	0x03	0x04		0x05	0x00	Output	—	[7:0] are 8 MSBs of 14-bit real-time sample
OUT_Y_LSB ⁽¹⁾⁽²⁾	R	0x04	0x05		0x00		Output	—	[7:2] are 6 LSBs of 14-bit real-time sample
OUT_Z_MSB ⁽¹⁾⁽²⁾	R	0x05	0x06		0x00		Output	—	[7:0] are 8 MSBs of 14-bit real-time sample
OUT_Z_LSB ⁽¹⁾⁽²⁾	R	0x06	0x00				Output	—	[7:2] are 6 LSBs of 14-bit real-time sample
Reserved	R	0x07	—	—	—	—	—	—	Reserved. Read return 0x00.
Reserved	R	0x08	—	—	—	—	—	—	Reserved. Read return 0x00.
F_SETUP ⁽¹⁾⁽³⁾	R/W	0x09	0x0A				00000000	0x00	FIFO setup
TRIG_CFG ⁽¹⁾⁽⁴⁾	R/W	0x0A	0x0B				00000000	0x00	Map of FIFO data capture events
SYSMOD ⁽¹⁾⁽²⁾	R	0x0B	0x0C				00000000	0x00	Current System Mode
INT_SOURCE ⁽¹⁾⁽²⁾	R	0x0C	0x0D				00000000	0x00	Interrupt status
WHO_AM_I ⁽¹⁾	R	0x0D	0x0E				00011010	0x1A	Device ID (0x1A)
XYZ_DATA_CFG ⁽¹⁾⁽⁴⁾	R/W	0x0E	0x0F				00000000	0x00	Dynamic Range Settings
HP_FILTER_CUTOFF ⁽¹⁾⁽⁴⁾	R/W	0x0F	0x10				00000000	0x00	Cutoff frequency is set to 16 Hz @ 800 Hz
PL_STATUS ⁽¹⁾⁽²⁾	R	0x10	0x11				00000000	0x00	Landscape/Portrait orientation status
PL_CFG ⁽¹⁾⁽⁴⁾	R/W	0x11	0x12				10000000	0x80	Landscape/Portrait configuration.
PL_COUNT ⁽¹⁾⁽³⁾	R/W	0x12	0x13				00000000	0x00	Landscape/Portrait debounce counter
PL_BF_ZCOMP ⁽¹⁾⁽⁴⁾	R/W	0x13	0x14				01000100	0x44	Back/Front, Z-Lock Trip threshold
P_L_THS_REG ⁽¹⁾⁽⁴⁾	R/W	0x14	0x15				10000100	0x84	Portrait to Landscape Trip Angle is 29°
FF_MT_CFG ⁽¹⁾⁽⁴⁾	R/W	0x15	0x16				00000000	0x00	Freefall/Motion functional block configuration
FF_MT_SRC ⁽¹⁾⁽²⁾	R	0x16	0x17				00000000	0x00	Freefall/Motion event source register
FF_MT_THS ⁽¹⁾⁽³⁾	R/W	0x17	0x18				00000000	0x00	Freefall/Motion threshold register
FF_MT_COUNT ⁽¹⁾⁽³⁾	R/W	0x18	0x19				00000000	0x00	Freefall/Motion debounce counter
Reserved	R	0x19	—	—	—	—	—	—	Reserved. Read return 0x00.
Reserved	R	0x1A	—	—	—	—	—	—	Reserved. Read return 0x00.
Reserved	R	0x1B	—	—	—	—	—	—	Reserved. Read return 0x00.
Reserved	R	0x1C	—	—	—	—	—	—	Reserved. Read return 0x00.
TRANSIENT_CFG ⁽¹⁾⁽⁴⁾	R/W	0x1D	0x1E				00000000	0x00	Transient functional block configuration
TRANSIENT_SCR ⁽¹⁾⁽²⁾	R	0x1E	0x1F				00000000	0x00	Transient event status register

MMA8451Q

Table 12. Register Address Map

TRANSIENT_THS ⁽¹⁾⁽³⁾	R/W	0x1F	0x20	00000000	0x00	Transient event threshold
TRANSIENT_COUNT ⁽¹⁾⁽³⁾	R/W	0x20	0x21	00000000	0x00	Transient debounce counter
PULSE_CFG ⁽¹⁾⁽⁴⁾	R/W	0x21	0x22	00000000	0x00	ELE, Double_XYZ or Single_XYZ
PULSE_SRC ⁽¹⁾⁽²⁾	R	0x22	0x23	00000000	0x00	EA, Double_XYZ or Single_XYZ
PULSE_THSX ⁽¹⁾⁽³⁾	R/W	0x23	0x24	00000000	0x00	X pulse threshold
PULSE_THSY ⁽¹⁾⁽³⁾	R/W	0x24	0x25	00000000	0x00	Y pulse threshold
PULSE_THSZ ⁽¹⁾⁽⁴⁾	R/W	0x25	0x26	00000000	0x00	Z pulse threshold
PULSE_TMLT ⁽¹⁾⁽⁴⁾	R/W	0x26	0x27	00000000	0x00	Time limit for pulse
PULSE_LTCY ⁽¹⁾⁽⁴⁾	R/W	0x27	0x28	00000000	0x00	Latency time for 2 nd pulse
PULSE_WIND ⁽¹⁾⁽⁴⁾	R/W	0x28	0x29	00000000	0x00	Window time for 2nd pulse
ASLP_COUNT ⁽¹⁾⁽⁴⁾	R/W	0x29	0x2A	00000000	0x00	Counter setting for Auto-SLEEP
CTRL_REG1 ⁽¹⁾⁽⁴⁾	R/W	0x2A	0x2B	00000000	0x00	ODR = 800 Hz, STANDBY Mode.
CTRL_REG2 ⁽¹⁾⁽⁴⁾	R/W	0x2B	0x2C	00000000	0x00	Sleep Enable, OS Modes, RST, ST
CTRL_REG3 ⁽¹⁾⁽⁴⁾	R/W	0x2C	0x2D	00000000	0x00	Wake from Sleep, IPOL, PP_OD
CTRL_REG4 ⁽¹⁾⁽⁴⁾	R/W	0x2D	0x2E	00000000	0x00	Interrupt enable register
CTRL_REG5 ⁽¹⁾⁽⁴⁾	R/W	0x2E	0x2F	00000000	0x00	Interrupt pin (INT1/INT2) map
OFF_X ⁽¹⁾⁽⁴⁾	R/W	0x2F	0x30	00000000	0x00	X-axis offset adjust
OFF_Y ⁽¹⁾⁽⁴⁾	R/W	0x30	0x31	00000000	0x00	Y-axis offset adjust
OFF_Z ⁽¹⁾⁽⁴⁾	R/W	0x31	0x0D	00000000	0x00	Z-axis offset adjust
Reserved (do not modify)		0x40 – 7F	—	—	—	Reserved. Read return 0x00.

1. Register contents are preserved when transition from ACTIVE to STANDBY mode occurs.
2. Register contents are reset when transition from STANDBY to ACTIVE mode occurs.
3. Register contents can be modified anytime in STANDBY or ACTIVE mode. A write to this register will cause a reset of the corresponding internal system debounce counter.
4. Modification of this register's contents can only occur when device is STANDBY mode except CTRL_REG1 ACTIVE bit and CTRL_REG2 RST bit.

Note: Auto-increment addresses which are not a simple increment are highlighted in **bold**. The auto-increment addressing is only enabled when device registers are read using I²C burst read mode. Therefore the internal storage of the auto-increment address is cleared whenever a stop-bit is detected.

6.1 Data Registers

The following are the data registers for the MMA8451Q. For more information on data manipulation of the MMA8451Q, refer to application note, AN4076.

When the F_MODE bits found in Register 0x09 (F_SETUP), bits 7 and 6 are both cleared (the FIFO is not on). Register 0x00 reflects the real-time status information of the X, Y and Z sample data. When the F_MODE value is greater than zero the FIFO is on (in either Fill, Circular or Trigger mode). In this case Register 0x00 will reflect the status of the FIFO. It is expected when the FIFO is on that the user will access the data from Register 0x01 (X_MSB) for either the 14-bit or 8-bit data. When accessing the 8-bit data the F_READ bit (Register 0x2A) is set which modifies the auto-incrementing to skip over the LSB data. When F_READ bit is cleared the 14-bit data is read accessing all 6 bytes sequentially (X_MSB, X_LSB, Y_MSB, Y_LSB, Z_MSB, Z_LSB).

F_MODE = 00: 0x00 STATUS: Data Status Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ZYXOW	ZOW	YOW	XOW	ZYXDR	ZDR	YDR	XDR

Table 13. STATUS Description

ZYXOW	X, Y, Z-axis Data Overwrite. Default value: 0 0: No data overwrite has occurred 1: Previous X, Y, or Z data was overwritten by new X, Y, or Z data before it was read
ZOW	Z-axis Data Overwrite. Default value: 0 0: No data overwrite has occurred 1: Previous Z-axis data was overwritten by new Z-axis data before it was read
YOW	Y-axis Data Overwrite. Default value: 0 0: No data overwrite has occurred 1: Previous Y-axis data was overwritten by new Y-axis data before it was read
XOW	X-axis Data Overwrite. Default value: 0 0: No data overwrite has occurred 1: Previous X-axis data was overwritten by new X-axis data before it was read
ZYXDR	X, Y, Z-axis new Data Ready. Default value: 0 0: No new set of data ready 1: A new set of data is ready
ZDR	Z-axis new Data Available. Default value: 0 0: No new Z-axis data is ready 1: A new Z-axis data is ready
YDR	Y-axis new Data Available. Default value: 0 0: No new Y-axis data ready 1: A new Y-axis data is ready
XDR	X-axis new Data Available. Default value: 0 0: No new X-axis data ready 1: A new X-axis data is ready

ZYXOW is set whenever a new acceleration data is produced before completing the retrieval of the previous set. This event occurs when the content of at least one acceleration data register (i.e., OUT_X, OUT_Y, OUT_Z) has been overwritten. ZYXOW is cleared when the high-bytes of the acceleration data (OUT_X_MSB, OUT_Y_MSB, OUT_Z_MSB) of all the active channels are read.

ZOW is set whenever a new acceleration sample related to the Z-axis is generated before the retrieval of the previous sample. When this occurs the previous sample is overwritten. ZOW is cleared anytime OUT_Z_MSB register is read.

YOW is set whenever a new acceleration sample related to the Y-axis is generated before the retrieval of the previous sample. When this occurs the previous sample is overwritten. YOW is cleared anytime OUT_Y_MSB register is read.

XOW is set whenever a new acceleration sample related to the X-axis is generated before the retrieval of the previous sample. When this occurs the previous sample is overwritten. XOW is cleared anytime OUT_X_MSB register is read.

ZYXDR signals that a new sample for any of the enabled channels is available. ZYXDR is cleared when the high-bytes of the acceleration data (OUT_X_MSB, OUT_Y_MSB, OUT_Z_MSB) of all the enabled channels are read.

ZDR is set whenever a new acceleration sample related to the Z-axis is generated. ZDR is cleared anytime OUT_Z_MSB register is read.

YDR is set whenever a new acceleration sample related to the Y-axis is generated. YDR is cleared anytime OUT_Y_MSB register is read.

XDR is set whenever a new acceleration sample related to the X-axis is generated. XDR is cleared anytime OUT_X_MSB register is read.

Data Registers: 0x01 OUT_X_MSB, 0x02 OUT_X_LSB, 0x03 OUT_Y_MSB, 0x04 OUT_Y_LSB, 0x05 OUT_Z_MSB, 0x06 OUT_Z_LSB

These registers contain the X-axis, Y-axis, and Z-axis 14-bit output sample data expressed as 2's complement numbers.

Note: The sample data output registers store the current sample data if the FIFO data output register driver is disabled, but if the FIFO data output register driver is enabled ($F_MODE > 00$) the sample data output registers point to the head of the FIFO buffer (Register 0x01 X_MSB) which contains the previous 32 X, Y, and Z data samples. Data Registers $F_MODE = 00$

0x01: OUT_X_MSB: X_MSB Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
XD13	XD12	XD11	XD10	XD9	XD8	XD7	XD6

0x02: OUT_X_LSB: X_LSB Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
XD5	XD4	XD3	XD2	XD1	XD0	0	0

0x03: OUT_Y_MSB: Y_MSB Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
YD13	YD12	YD11	YD10	YD9	YD8	YD7	YD6

0x04: OUT_Y_LSB: Y_LSB Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
YD5	YD4	YD3	YD2	YD1	YD0	0	0

0x05: OUT_Z_MSB: Z_MSB Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ZD13	ZD12	ZD11	ZD10	ZD9	ZD8	ZD7	ZD6

0x06: OUT_Z_LSB: Z_LSB Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ZD5	ZD4	ZD3	ZD2	ZD1	ZD0	0	0

OUT_X_MSB, OUT_X_LSB, OUT_Y_MSB, OUT_Y_LSB, OUT_Z_MSB, and OUT_Z_LSB are stored in the auto-incrementing address range of 0x01 to 0x06 to reduce reading the status followed by 14-bit axis data to 7 bytes. If the F_READ bit is set (0x2A bit 1), auto increment will skip over LSB registers. This will shorten the data acquisition from 7 bytes to 4 bytes. The LSB registers can only be read immediately following the read access of the corresponding MSB register. A random read access to the LSB registers is not possible. Reading the MSB register and then the LSB register in sequence ensures that both bytes (LSB and MSB) belong to the same data sample, even if a new data sample arrives between reading the MSB and the LSB byte.

6.2 32 Sample FIFO

The following registers are used to configure the FIFO. For more information on the FIFO please refer to AN4073.

F_MODE > 0 0x00: F_STATUS FIFO Status Register

When $F_MODE > 0$, Register 0x00 becomes the FIFO Status Register which is used to retrieve information about the FIFO. This register has a flag for the overflow and watermark. It also has a counter that can be read to obtain the number of samples stored in the buffer when the FIFO is enabled.

0x00: F_STATUS: FIFO STATUS Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F_OVF	F_WMRK_FLAG	F_CNT5	F_CNT4	F_CNT3	F_CNT2	F_CNT1	F_CNT0

Table 14. FIFO Flag Event Description

F_OVF	F_WMRK_FLAG	Event Description
0	—	No FIFO overflow events detected.
1	—	FIFO event detected; FIFO has overflowed.
—	0	No FIFO watermark events detected.
—	1	FIFO Watermark event detected. FIFO sample count is greater than watermark value. If F_MODE = 11, Trigger Event detected.

The F_OVF and F_WMRK_FLAG flags remain asserted while the event source is still active, but the user can clear the FIFO interrupt bit flag in the interrupt source register (INT_SOURCE) by reading the F_STATUS register. In this case, the SRC_FIFO bit in the INT_SOURCE register will be set again when the next data sample enters the FIFO. Therefore the F_OVF bit flag will remain asserted while the FIFO has overflowed and the F_WMRK_FLAG bit flag will remain asserted while the F_CNT value is equal to or greater than then F_WMRK value. If the FIFO overflow flag is cleared and if F_MODE = 11 then the FIFO overflow flag will remain 0 before the trigger event even if the FIFO is full and overflows. If the FIFO overflow flag is set and if F_MODE = 11, the FIFO has stopped accepting samples.

Table 15. FIFO Sample Count Description

F_CNT[5:0]	FIFO sample counter. Default value: 00_0000. (00_0001 to 10_0000 indicates 1 to 32 samples stored in FIFO)
------------	---

F_CNT[5:0] bits indicate the number of acceleration samples currently stored in the FIFO buffer. Count 000000 indicates that the FIFO is empty.

0x09: F_SETUP FIFO Setup Register

0x09 F_SETUP: FIFO Setup Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F_MODE1	F_MODE0	F_WMRK5	F_WMRK4	F_WMRK3	F_WMRK2	F_WMRK1	F_WMRK0

Table 16. F_SETUP Description

BITS	Description
F_MODE[1:0] ⁽¹⁾⁽²⁾	FIFO buffer overflow mode. Default value: 0. 00: FIFO is disabled. 01: FIFO contains the most recent samples when overflowed (circular buffer). Oldest sample is discarded to be replaced by new sample. 10: FIFO stops accepting new samples when overflowed. 11: Trigger mode. The FIFO will be in a circular mode up to the number of samples in the watermark. The FIFO will be in a circular mode until the trigger event occurs after that the FIFO will continue to accept samples for 32-WMRK samples and then stop receiving further samples. This allows data to be collected both before and after the trigger event and it is definable by the watermark setting. The FIFO is flushed whenever the FIFO is disabled, during an automatic ODR change (Auto-WAKE/SLEEP), or transitioning from STANDBY mode to ACTIVE mode. Disabling the FIFO (F_MODE = 00) resets the F_OVF, F_WMRK_FLAG, F_CNT to zero. A FIFO overflow event (i.e., F_CNT = 32) will assert the F_OVF flag and a FIFO sample count equal to the sample count watermark (i.e., F_WMRK) asserts the F_WMRK_FLAG event flag.
F_WMRK[5:0] ⁽²⁾	FIFO Event Sample Count Watermark. Default value: 00_0000. These bits set the number of FIFO samples required to trigger a watermark interrupt. A FIFO watermark event flag is raised when FIFO sample count F_CNT[5:0] ≥ F_WMRK[5:0] watermark. Setting the F_WMRK[5:0] to 00_0000 will disable the FIFO watermark event flag generation. Also used to set the number of pre-trigger samples in Trigger mode.

1. Bit field can be written in ACTIVE mode.
2. Bit field can be written in STANDBY mode.

The FIFO mode can be changed while in the active state. The mode must first be disabled F_MODE = 00 then the mode can be switched between Fill mode, Circular mode and Trigger mode.

A FIFO sample count exceeding the watermark event does not stop the FIFO from accepting new data. The FIFO update rate is dictated by the selected system ODR. In ACTIVE mode the ODR is set by the DR bits in the CTRL_REG1 register. When Auto-SLEEP is active the ODR is set by the ASLP_RATE field in the CTRL_REG1 register.

When a byte is read from the FIFO buffer the oldest sample data in the FIFO buffer is returned and also deleted from the front of the FIFO buffer, while the FIFO sample count is decremented by one. It is assumed that the host application shall use the I²C multi-byte read transaction to empty the FIFO.

0x0A: TRIG_CFG

In the trigger configuration register the bits that are set (logic '1') control which function may trigger the FIFO to its interrupt and conversely bits that are cleared (logic '0') indicate which function has not asserted its interrupt.

The bits set are rising edge sensitive, and are set by a low to high state change and reset by reading the appropriate source register.

0x0A: TRIG_CFG Trigger Configuration Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
—	—	Trig_TRANS	Trig_LNDPRT	Trig_PULSE	Trig_FF_MT	—	—

Table 17. Trigger Configuration Description

INT_SOURCE	Description
Trig_TRANS	Transient interrupt trigger bit. Default value: 0
Trig_LNDPRT	Landscape/Portrait Orientation interrupt trigger bit. Default value: 0
Trig_PULSE	Pulse interrupt trigger bit. Default value: 0
Trig_FF_MT	Freefall/Motion trigger bit. Default value: 0

0x0B: SYSMOD System Mode Register

The System mode register indicates the current device operating mode. Applications using the Auto-SLEEP/WAKE mechanism should use this register to synchronize the application with the device operating mode transitions. The System mode register also indicates the status of the FIFO gate error and number of samples since the gate error occurred.

0x0B: SYSMOD: System Mode Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FGERR	FGT_4	FGT_3	FGT_2	FGT_1	FGT_0	SYSMOD1	SYSMOD0

Table 18. SYSMOD Description

FGERR	FIFO Gate Error. Default value: 0. 0: No FIFO Gate Error detected. 1: FIFO Gate Error was detected. Emptying the FIFO buffer clears the FGERR bit in the SYS_MOD register. See section 0x2C: CTRL_REG3 Interrupt Control Register for more information on configuring the FIFO Gate function.
FGT[4:0]	Number of ODR time units since FGERR was asserted. Reset when FGERR Cleared. Default value: 0_0000
SYSMOD[1:0]	System Mode. Default value: 00. 00: STANDBY mode 01: WAKE mode 10: SLEEP mode

0x0C: INT_SOURCE System Interrupt Status Register

In the interrupt source register the status of the various embedded features can be determined. The bits that are set (logic '1') indicate which function has asserted an interrupt and conversely the bits that are cleared (logic '0') indicate which function has not asserted or has deasserted an interrupt. **The bits are set by a low to high transition and are cleared by reading the appropriate interrupt source register.** The SRC_DRDY bit is cleared by reading the X, Y and Z data. It is not cleared by simply reading the Status Register (0x00).

0x0C: INT_SOURCE: System Interrupt Status Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SRC_ASLP	SRC_FIFO	SRC_TRANS	SRC_LNDPRT	SRC_PULSE	SRC_FF_MT	—	SRC_DRDY

Table 19. INT_SOURCE Description

INT_SOURCE	Description
SRC_ASLP	Auto-SLEEP/WAKE interrupt status bit. Default value: 0. Logic '1' indicates that an interrupt event that can cause a WAKE to SLEEP or SLEEP to WAKE system mode transition has occurred. Logic '0' indicates that no WAKE to SLEEP or SLEEP to WAKE system mode transition interrupt event has occurred. WAKE to SLEEP transition occurs when no interrupt occurs for a time period that exceeds the user specified limit (ASLP_COUNT). This causes the system to transition to a user specified low ODR setting. SLEEP to WAKE transition occurs when the user specified interrupt event has woken the system; thus causing the system to transition to a user specified high ODR setting. Reading the SYSMOD register clears the SRC_ASLP bit.
SRC_FIFO	FIFO interrupt status bit. Default value: 0. Logic '1' indicates that a FIFO interrupt event such as an overflow event or watermark has occurred. Logic '0' indicates that no FIFO interrupt event has occurred. FIFO interrupt event generators: FIFO Overflow, or (Watermark: F_CNT = F_WMRK) and the interrupt has been enabled. This bit is cleared by reading the F_STATUS register.
SRC_TRANS	Transient interrupt status bit. Default value: 0. Logic '1' indicates that an acceleration transient value greater than user specified threshold has occurred. Logic '0' indicates that no transient event has occurred. This bit is asserted whenever "EA" bit in the TRANS_SRC is asserted and the interrupt has been enabled. This bit is cleared by reading the TRANS_SRC register.
SRC_LNDPRT	Landscape/Portrait Orientation interrupt status bit. Default value: 0. Logic '1' indicates that an interrupt was generated due to a change in the device orientation status. Logic '0' indicates that no change in orientation status was detected. This bit is asserted whenever "NEWLP" bit in the PL_STATUS is asserted and the interrupt has been enabled. This bit is cleared by reading the PL_STATUS register.
SRC_PULSE	Pulse interrupt status bit. Default value: 0. Logic '1' indicates that an interrupt was generated due to single and/or double pulse event. Logic '0' indicates that no pulse event was detected. This bit is asserted whenever "EA" bit in the PULSE_SRC is asserted and the interrupt has been enabled. This bit is cleared by reading the PULSE_SRC register.
SRC_FF_MT	Freefall/Motion interrupt status bit. Default value: 0. Logic '1' indicates that the Freefall/Motion function interrupt is active. Logic '0' indicates that no Freefall or Motion event was detected. This bit is asserted whenever "EA" bit in the FF_MT_SRC register is asserted and the FF_MT interrupt has been enabled. This bit is cleared by reading the FF_MT_SRC register.
SRC_DRDY	Data Ready Interrupt bit status. Default value: 0. Logic '1' indicates that the X, Y, Z data ready interrupt is active indicating the presence of new data and/or data overrun. Otherwise if it is a logic '0' the X, Y, Z interrupt is not active. This bit is asserted when the ZYXOW and/or ZYXDR is set and the interrupt has been enabled. This bit is cleared by reading the X, Y, and Z data.

0x0D: WHO_AM_I Device ID Register

The device identification register identifies the part. The default value is 0x1A. This value is factory programmed. Consult the factory for custom alternate values.

0x0D: WHO_AM_I Device ID Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	1	1	0	1	0

0x0E: XYZ_DATA_CFG Register

The XYZ_DATA_CFG register sets the dynamic range and sets the high-pass filter for the output data. When the HPF_OUT bit is set, both the FIFO and DATA registers will contain high-pass filtered data.

0x0E: XYZ_DATA_CFG (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	HPF_OUT	0	0	FS1	FS0

Table 20. XYZ Data Configuration Descriptions

HPF_OUT	Enable High-pass output data 1 = output data High-pass filtered. Default value: 0.
FS[1:0]	Output buffer data format full scale. Default value: 00 (2g).

The default full scale value range is 2g and the high-pass filter is disabled.

Table 21. Full Scale Range

FS1	FS0	Full Scale Range
0	0	2
0	1	4
1	0	8
1	1	Reserved

0x0F: HP_FILTER_CUTOFF High-Pass Filter Register

This register sets the high-pass filter cutoff frequency for removal of the offset and slower changing acceleration data. The output of this filter is indicated by the data registers (0x01-0x06) when bit 4 (HPF_OUT) of Register 0x0E is set. The filter cutoff options change based on the data rate selected as shown in [Table 23](#). For details of implementation on the high-pass filter, refer to Freescale application note, AN4071.

0x0F: HP_FILTER_CUTOFF: High-Pass Filter Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	Pulse_HPF_BYP	Pulse_LPF_EN	0	0	SEL1	SEL0

Table 22. High-Pass Filter Cutoff Register Descriptions

Pulse_HPF_BYP	Bypass High-Pass Filter (HPF) for Pulse Processing Function. 0: HPF enabled for Pulse Processing, 1: HPF Bypassed for Pulse Processing Default value: 0.
Pulse_LPF_EN	Enable Low-Pass Filter (LPF) for Pulse Processing Function. 0: LPF disabled for Pulse Processing, 1: LPF Enabled for Pulse Processing Default value: 0.
SEL[1:0]	HPF Cutoff frequency selection. Default value: 00 (see Table 23).

Table 23. High-Pass Filter Cutoff Options

Oversampling Mode = Normal									
SEL1	SEL0	800 Hz	400 Hz	200 Hz	100 Hz	50 Hz	12.5 Hz	6.25 Hz	1.56 Hz
0	0	16 Hz	16 Hz	8 Hz	4 Hz	2 Hz	2 Hz	2 Hz	2 Hz
0	1	8 Hz	8 Hz	4 Hz	2 Hz	1 Hz	1 Hz	1 Hz	1 Hz
1	0	4 Hz	4 Hz	2 Hz	1 Hz	0.5 Hz	0.5 Hz	0.5 Hz	0.5 Hz
1	1	2 Hz	2 Hz	1 Hz	0.5 Hz	0.25 Hz	0.25 Hz	0.25 Hz	0.25 Hz
Oversampling Mode = Low Noise Low Power									
0	0	16 Hz	16 Hz	8 Hz	4 Hz	2 Hz	0.5 Hz	0.5 Hz	0.5 Hz
0	1	8 Hz	8 Hz	4 Hz	2 Hz	1 Hz	0.25 Hz	0.25 Hz	0.25 Hz
1	0	4 Hz	4 Hz	2 Hz	1 Hz	0.5 Hz	0.125 Hz	0.125 Hz	0.125 Hz
1	1	2 Hz	2 Hz	1 Hz	0.5 Hz	0.25 Hz	0.063 Hz	0.063 Hz	0.063 Hz
Oversampling Mode = High Resolution									
0	0	16 Hz	16 Hz	16 Hz	16 Hz	16 Hz	16 Hz	16 Hz	16 Hz
0	1	8 Hz	8 Hz	8 Hz	8 Hz	8 Hz	8 Hz	8 Hz	8 Hz
1	0	4 Hz	4 Hz	4 Hz	4 Hz	4 Hz	4 Hz	4 Hz	4 Hz
1	1	2 Hz	2 Hz	2 Hz	2 Hz	2 Hz	2 Hz	2 Hz	2 Hz
Oversampling Mode = Low Power									
0	0	16 Hz	8 Hz	4 Hz	2 Hz	1 Hz	0.25 Hz	0.25 Hz	0.25 Hz
0	1	8 Hz	4 Hz	2 Hz	1 Hz	0.5 Hz	0.125 Hz	0.125 Hz	0.125 Hz
1	0	4 Hz	2 Hz	1 Hz	0.5 Hz	0.25 Hz	0.063 Hz	0.063 Hz	0.063 Hz
1	1	2 Hz	1 Hz	0.5 Hz	0.25 Hz	0.125 Hz	0.031 Hz	0.031 Hz	0.031 Hz

6.3 Portrait/Landscape Embedded Function Registers

For more details on the meaning of the different user configurable settings and for example code refer to Freescale application note, AN4068.

0x10: PL_STATUS Portrait/Landscape Status Register

This status register can be read to get updated information on any change in orientation by reading Bit 7, or on the specifics of the orientation by reading the other bits. For further understanding of Portrait Up, Portrait Down, Landscape Left, Landscape Right, Back and Front orientations please refer to [Figure 3](#). The interrupt is cleared when reading the PL_STATUS register.

0x10: PL_STATUS Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
NEWLP	LO	—	—	—	LAPO[1]	LAPO[0]	BAFRO

Table 24. PL_STATUS Register Description

NEWLP	Landscape/Portrait status change flag. Default value: 0. 0: No change, 1: BAFRO and/or LAPO and/or Z-Tilt lockout value has changed
LO	Z-Tilt Angle Lockout. Default value: 0. 0: Lockout condition has not been detected. 1: Z-Tilt lockout trip angle has been exceeded. Lockout has been detected.
LAPO[1:0] ⁽¹⁾	Landscape/Portrait orientation. Default value: 00 00: Portrait Up: Equipment standing vertically in the normal orientation 01: Portrait Down: Equipment standing vertically in the inverted orientation 10: Landscape Right: Equipment is in landscape mode to the right 11: Landscape Left: Equipment is in landscape mode to the left.
BAFRO	Back or Front orientation. Default value: 0 0: Front: Equipment is in the front facing orientation. 1: Back: Equipment is in the back facing orientation.

1. The default power up state is BAFRO = 0, LAPO = 0, and LO = 0.

NEWLP is set to 1 after the first orientation detection after a STANDBY to ACTIVE transition, and whenever a change in LO, BAFRO, or LAPO occurs. NEWLP bit is cleared anytime PL_STATUS register is read. The Orientation mechanism state change is limited to a maximum 1.25g. LAPO BAFRO and LO continue to change when NEWLP is set. The current position is locked if the absolute value of the acceleration experienced on any of the three axes is greater than 1.25g.

0x11: Portrait/Landscape Configuration Register

This register enables the Portrait/Landscape function and sets the behavior of the debounce counter.

0x11: PL_CFG Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBCNTM	PL_EN	—	—	—	—	—	—

Table 25. PL_CFG Description

DBCNTM	Debounce counter mode selection. Default value: 1 0: Decrements debounce whenever condition of interest is no longer valid. 1: Clears counter whenever condition of interest is no longer valid.
PL_EN	Portrait/Landscape Detection Enable. Default value: 0 0: Portrait/Landscape Detection is Disabled. 1: Portrait/Landscape Detection is Enabled.

0x12: Portrait/Landscape Debounce Counter

This register sets the debounce count for the orientation state transition. The minimum debounce latency is determined by the data rate set by the product of the selected system ODR and PL_COUNT registers. Any transition from WAKE to SLEEP or vice versa resets the internal Landscape/Portrait debounce counter. **Note:** The debounce counter weighting (time step) changes based on the ODR and the Oversampling mode. Table 27 explains the time step value for all sample rates and all Oversampling modes.

0x12: PL_COUNT Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBNCE[7]	DBNCE[6]	DBNCE[5]	DBNCE[4]	DBNCE[3]	DBNCE[2]	DBNCE[1]	DBNCE[0]

Table 26. PL_COUNT Description

DBNCE[7:0]	Debounce Count value. Default value: 0000_0000.
------------	---

Table 27. PL_COUNT Relationship with the ODR

ODR (Hz)	Max Time Range (s)				Time Step (ms)			
	Normal	LPLN	HighRes	LP	Normal	LPLN	HighRes	LP
800	0.319	0.319	0.319	0.319	1.25	1.25	1.25	1.25
400	0.638	0.638	0.638	0.638	2.5	2.5	2.5	2.5
200	1.28	1.28	0.638	1.28	5	5	2.5	5
100	2.55	2.55	0.638	2.55	10	10	2.5	10
50	5.1	5.1	0.638	5.1	20	20	2.5	20
12.5	5.1	20.4	0.638	20.4	20	80	2.5	80
6.25	5.1	20.4	0.638	40.8	20	80	2.5	160
1.56	5.1	20.4	0.638	40.8	20	80	2.5	160

0x13: PL_BF_ZCOMP Back/Front and Z Compensation Register

The Z-Lock angle compensation bits allow the user to adjust the Z-lockout region from 14° up to 43°. The default Z-lockout angle is set to the default value of 29° upon power up. The Back to Front trip angle is set by default to ±75° but this angle also can be adjusted from a range of 65° to 80° with 5° step increments.

0x13: PL_BF_ZCOMP Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BKFR[1]	BKFR[0]	—	—	—	ZLOCK[2]	ZLOCK[1]	ZLOCK[0]

Table 28. PL_BF_ZCOMP Description

BKFR[7:6]	Back/Front Trip Angle Threshold. Default: 01 ≥ ±75°. Step size is 5°. Range: ±(65° to 80°).
ZLOCK[2:0]	Z-Lock Angle Threshold. Range is from 14° to 43°. Step size is 4°. Default value: 100 ≥ 29°. Maximum value: 111 ≥ 43°.

Note: All angles are accurate to ±2°.

Table 29. Z-Lock Threshold Angles

Z-Lock Value	Threshold Angle
0x00	14°
0x01	18°
0x02	21°
0x03	25°
0x04	29°
0x05	33°
0x06	37°
0x07	42°

Table 30. Back/Front Orientation Definition

BKFR	Back/Front Transition	Front/Back Transition
00	Z < 80° or Z > 280°	Z > 100° and Z < 260°
01	Z < 75° or Z > 285°	Z > 105° and Z < 255°
10	Z < 70° or Z > 290°	Z > 110° and Z < 250°
11	Z < 65° or Z > 295°	Z > 115° and Z < 245°

0x14: P_L_THS_REG Portrait/Landscape Threshold and Hysteresis Register

This register represents the Portrait to Landscape trip threshold register used to set the trip angle for transitioning from Portrait to Landscape and Landscape to Portrait. This register includes a value for the hysteresis.

0x14: P_L_THS_REG Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P_L_THS[4]	P_L_THS[3]	P_L_THS[2]	P_L_THS[1]	P_L_THS[0]	HYS[2]	HYS[1]	HYS[0]

Table 31. P_L_THS_REG Description

P_L_THS[7:3]	Portrait/Landscape trip threshold angle from 15° to 75°. See Table 32 for the values with the corresponding approximate threshold angle. Default value: 1_0000 (45°).
HYS[2:0]	This angle is added to the threshold angle for a smoother transition from Portrait to Landscape and Landscape to Portrait. This angle ranges from 0° to ±24°. The default is 100 (±14°).

[Table 32](#) is a lookup table to set the threshold. This is the center value that will be set for the trip point from Portrait to Landscape and Landscape to Portrait. The default Trip Angle is 45° (0x10). The default hysteresis is ±14°.

Note: The condition THS + HYS > 0 and THS + HYS < 32 must be met in order for Landscape/Portrait detection to work properly. The value of 32 represents the sum of both P_L_THS and HYS register values in decimal. For example, THS angle = 75°, P_L_THS = 25(dec) then max HYS must be set to 6 to meet the condition THS+HYS < 32. To configure correctly the hysteresis (HYS) angle must be smaller than the threshold angle (P_L_THS).

Table 32. Threshold Angle Thresholds Lookup Table

Threshold Angle (approx.)	5-bit Register value
15°	0x07
20°	0x09
30°	0x0C
35°	0x0D
40°	0x0F
45°	0x10
55°	0x13
60°	0x14
70°	0x17
75°	0x19

Table 33. Trip Angles with Hysteresis for 45° Angle

Hysteresis Register Value	Hysteresis ± Angle Range	Landscape to Portrait Trip Angle	Portrait to Landscape Trip Angle
0	±0	45°	45°
1	±4	49°	41°
2	±7	52°	38°
3	±11	56°	34°
4	±14	59°	31°

Table 33. Trip Angles with Hysteresis for 45° Angle

5	±17	62°	28°
6	±21	66°	24°
7	±24	69°	21°

6.4 Motion and Freefall Embedded Function Registers

The freefall/motion function can be configured in either Freefall or Motion Detection mode via the **OAE** configuration bit (0x15 bit 6). The freefall/motion detection block can be disabled by setting all three bits ZEFE, YEFE, and XEFE to zero.

Depending on the register bits **ELE** (0x15 bit 7) and **OAE** (0x15 bit 6), each of the freefall and motion detection block can operate in four different modes:

Mode 1: Freefall Detection with ELE = 0, OAE = 0

In this mode, the **EA** bit (0x16 bit 7) indicates a freefall event after the debounce counter is complete. The ZEFE, YEFE, and XEFE control bits determine which axes are considered for the freefall detection. Once the EA bit is set, and DBCNTM = 0, the EA bit can get cleared only after the delay specified by FF_MT_COUNT. This is because the counter is in decrement mode. If DBCNTM = 1, the EA bit is cleared as soon as the freefall condition disappears, and will not be set again before the delay specified by FF_MT_COUNT has passed. Reading the FF_MT_SRC register does not clear the EA bit. The event flags (0x16) ZHE, ZHP, YHE, YHP, XHE, and XHP reflect the motion detection status (i.e. high g event) without any debouncing, provided that the corresponding bits ZEFE, YEFE, and/or XEFE are set.

Mode 2: Freefall Detection with ELE = 1, OAE = 0

In this mode, the **EA** event bit indicates a freefall event after the debounce counter. Once the debounce counter reaches the time value for the set threshold, the EA bit is set, and remains set until the FF_MT_SRC register is read. When the FF_MT_SRC register is read, the EA bit and the debounce counter are cleared and a new event can only be generated after the delay specified by FF_MT_CNT. The ZEFE, YEFE, and XEFE control bits determine which axes are considered for the freefall detection. While EA = 0, the event flags ZHE, ZHP, YHE, YHP, XHE, and XHP reflect the motion detection status (i.e., high g event) without any debouncing, provided that the corresponding bits ZEFE, YEFE, and/or XEFE are set. The event flags ZHE, ZHP, YHE, YHP, XHE, and XHP are latched when the EA event bit is set. The event flags ZHE, ZHP, YHE, YHP, XHE, and XHP will start changing only after the FF_MT_SRC register has been read.

Mode 3: Motion Detection with ELE = 0, OAE = 1

In this mode, the **EA** bit indicates a motion event after the debounce counter time is reached. The ZEFE, YEFE, and XEFE control bits determine which axes are taken into consideration for motion detection. Once the EA bit is set, and DBCNTM = 0, the EA bit can get cleared only after the delay specified by FF_MT_COUNT. If DBCNTM = 1, the EA bit is cleared as soon as the motion high g condition disappears. The event flags ZHE, ZHP, YHE, YHP, XHE, and XHP reflect the motion detection status (i.e., high g event) without any debouncing, provided that the corresponding bits ZEFE, YEFE, and/or XEFE are set. Reading the FF_MT_SRC does not clear any flags, nor is the debounce counter reset.

Mode 4: Motion Detection with ELE = 1, OAE = 1

In this mode, the EA bit indicates a motion event after debouncing. The ZEFE, YEFE, and XEFE control bits determine which axes are taken into consideration for motion detection. Once the debounce counter reaches the threshold, the EA bit is set, and remains set until the FF_MT_SRC register is read. When the FF_MT_SRC register is read, all register bits are cleared and the debounce counter are cleared and a new event can only be generated after the delay specified by FF_MT_CNT. While the bit EA is zero, the event flags ZHE, ZHP, YHE, YHP, XHE, and XHP reflect the motion detection status (i.e., high g event) without any debouncing, provided that the corresponding bits ZEFE, YEFE, and/or XEFE are set. When the EA bit is set, these bits keep their current value until the FF_MT_SRC register is read.

0x15: FF_MT_CFG Freefall/Motion Configuration Register

This is the Freefall/Motion configuration register for setting up the conditions of the freefall or motion function.

0x15: FF_MT_CFG Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ELE	OAE	ZEFE	YEFE	XEFE	—	—	—

Table 34. FF_MT_CFG Description

ELE	Event Latch Enable: Event flags are latched into FF_MT_SRC register. Reading of the FF_MT_SRC register clears the event flag EA and all FF_MT_SRC bits. Default value: 0. 0: Event flag latch disabled; 1: event flag latch enabled
OAE	Motion detect / Freefall detect flag selection. Default value: 0. (Freefall Flag) 0: Freefall Flag (Logical AND combination) 1: Motion Flag (Logical OR combination)
ZEFE	Event flag enable on Z Default value: 0. 0: event detection disabled; 1: raise event flag on measured acceleration value beyond preset threshold
YEFE	Event flag enable on Y event. Default value: 0. 0: Event detection disabled; 1: raise event flag on measured acceleration value beyond preset threshold
XEFE	Event flag enable on X event. Default value: 0. 0: event detection disabled; 1: raise event flag on measured acceleration value beyond preset threshold

OAE bit allows the selection between Motion (logical OR combination) and Freefall (logical AND combination) detection.

ELE denotes whether the enabled event flag will to be latched in the FF_MT_SRC register or the event flag status in the FF_MT_SRC will indicate the real-time status of the event. If ELE bit is set to a logic '1', then the event flags are frozen when the EA bit gets set, and are cleared by reading the FF_MT_SRC source register.

ZHFE, YEFE, XEFE enable the detection of a motion or freefall event when the measured acceleration data on X, Y, Z channel is beyond the threshold set in FF_MT_THS register. If the ELE bit is set to logic '1' in the FF_MT_CFG register new event flags are blocked from updating the FF_MT_SRC register.

FF_MT_THS is the threshold register used to detect freefall motion events. The unsigned 7-bit **FF_MT_THS** threshold register holds the threshold for the freefall detection where the magnitude of the X and Y and Z acceleration values is lower or equal than the threshold value. Conversely, the **FF_MT_THS** also holds the threshold for the motion detection where the magnitude of the X or Y or Z acceleration value is higher than the threshold value.

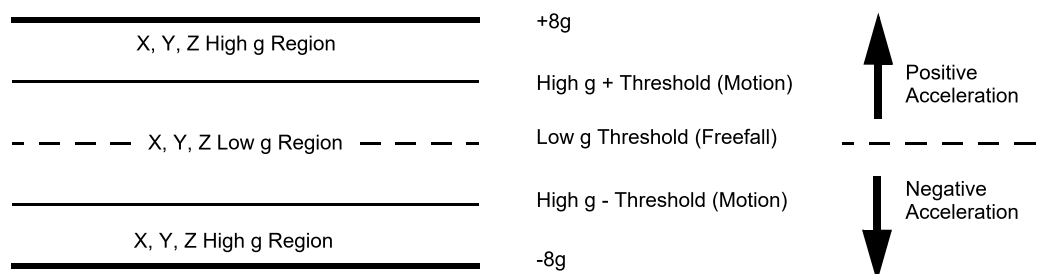


Figure 12. FF_MT_CFG High and Low g Level

0x16: FF_MT_SRC Freefall/Motion Source Register

0x16: FF_MT_SRC Freefall and Motion Source Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EA	—	ZHE	ZHP	YHE	YHP	XHE	XHP

Table 35. Freefall/Motion Source Description

EA	Event Active Flag. Default value: 0. 0: No event flag has been asserted; 1: one or more event flag has been asserted. See the description of the OAE bit to determine the effect of the 3-axis event flags on the EA bit.
ZHE	Z Motion Flag. Default value: 0. 0: No Z Motion event detected, 1: Z Motion has been detected This bit reads always zero if the ZEFE control bit is set to zero
ZHP	Z Motion Polarity Flag. Default value: 0. 0: Z event was Positive g, 1: Z event was Negative g This bit read always zero if the ZEFE control bit is set to zero
YHE	Y Motion Flag. Default value: 0. 0: No Y Motion event detected, 1: Y Motion has been detected This bit read always zero if the YEFE control bit is set to zero
YHP	Y Motion Polarity Flag. Default value: 0 0: Y event detected was Positive g, 1: Y event was Negative g This bit reads always zero if the YEFE control bit is set to zero
XHE	X Motion Flag. Default value: 0 0: No X Motion event detected, 1: X Motion has been detected This bit reads always zero if the XEFE control bit is set to zero
XHP	X Motion Polarity Flag. Default value: 0 0: X event was Positive g, 1: X event was Negative g This bit reads always zero if the XEFE control bit is set to zero

This register keeps track of the acceleration event which is triggering (or has triggered, in case of ELE bit in FF_MT_CFG register being set to 1) the event flag. In particular EA is set to a logic '1' when the logical combination of acceleration events flags specified in FF_MT_CFG register is true. This bit is used in combination with the values in INT_EN_FF_MT and INT_CFG_FF_MT register bits to generate the freefall/motion interrupts.

An X,Y, or Z motion is true when the acceleration value of the X or Y or Z channel is higher than the preset threshold value defined in the FF_MT_THS register.

Conversely an X, Y, and Z low event is true when the acceleration value of the X and Y and Z channel is lower than or equal to the preset threshold value defined in the FF_MT_THS register.

0x17: FF_MT_THS Freefall and Motion Threshold Register

0x17: FF_MT_THS Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBCNTM	THS6	THS5	THS4	THS3	THS2	THS1	THS0

Table 36. FF_MT_THS Description

DBCNTM	Debounce counter mode selection. Default value: 0. 0: increments or decrements debounce, 1: increments or clears counter.
THS[6:0]	Freefall /Motion Threshold: Default value: 000_0000.

The threshold resolution is 0.063g/LSB and the threshold register has a range of 0 to 127 counts. The maximum range is to 8g. Note that even when the full scale value is set to 2g or 4g the motion detects up to 8g. If the Low Noise bit is set in Register 0x2A then the maximum threshold will be limited to 4g regardless of the full scale range.

DBCNTM bit configures the way in which the debounce counter is reset when the inertial event of interest is momentarily not true.

When DBCNTM bit is a logic '1', the debounce counter is cleared to 0 whenever the inertial event of interest is no longer true as shown in Figure 13, (b). While the DBCNTM bit is set to logic '0' the debounce counter is decremented by 1 whenever the inertial event of interest is no longer true (Figure 13, (c)) until the debounce counter reaches 0 or the inertial event of interest becomes active.

Decrementing the debounce counter acts as a median enabling the system to filter out irregular spurious events which might impede the detection of inertial events.

0x18: FF_MT_COUNT Debounce Register

This register sets the number of debounce sample counts for the event trigger.

0x18: FF_MT_COUNT_Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

Table 37. FF_MT_COUNT Description

D[7:0]	Count value. Default value: 0000_0000
--------	---------------------------------------

This register sets the minimum number of debounce sample counts of continuously matching the detection condition user selected for the freefall, motion event.

When the internal debounce counter reaches the FF_MT_COUNT value a Freefall/Motion event flag is set. The debounce counter will never increase beyond the FF_MT_COUNT value. Time step used for the debounce sample count depends on the ODR chosen and the Oversampling mode as shown in [Table 38](#).

Table 38. FF_MT_COUNT Relationship with the ODR

ODR (Hz)	Max Time Range (s)				Time Step (ms)			
	Normal	LPLN	HighRes	LP	Normal	LPLN	HighRes	LP
800	0.319	0.319	0.319	0.319	1.25	1.25	1.25	1.25
400	0.638	0.638	0.638	0.638	2.5	2.5	2.5	2.5
200	1.28	1.28	0.638	1.28	5	5	2.5	5
100	2.55	2.55	0.638	2.55	10	10	2.5	10
50	5.1	5.1	0.638	5.1	20	20	2.5	20
12.5	5.1	20.4	0.638	20.4	20	80	2.5	80
6.25	5.1	20.4	0.638	40.8	20	80	2.5	160
1.56	5.1	20.4	0.638	40.8	20	80	2.5	160

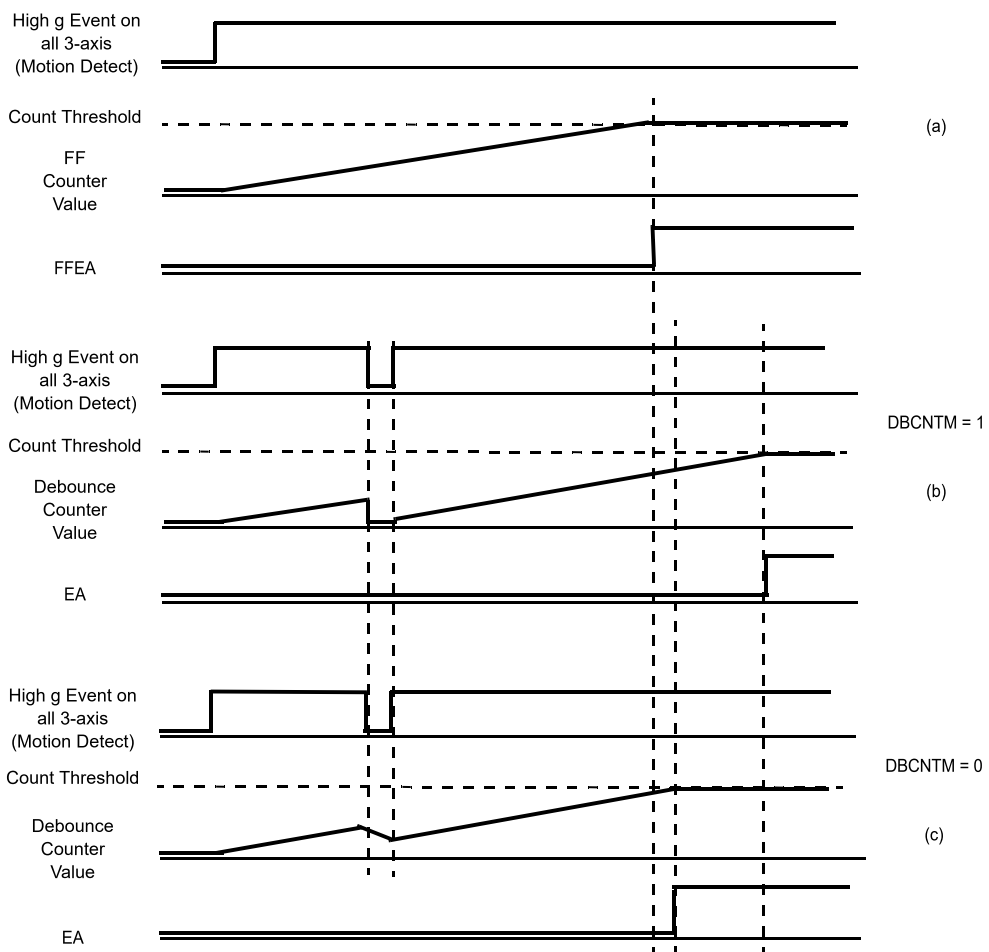


Figure 13. DBCNTM Bit Function

6.5 Transient (HPF) Acceleration Detection

For more information on the uses of the transient function please review Freescale application note, AN4071. This function is similar to the motion detection except that high-pass filtered data is compared. There is an option to disable the high-pass filter through the function. In this case the behavior is the same as the motion detection. This allows for the device to have 2 motion detection functions.

0x1D: Transient_CFG Register

The transient detection mechanism can be configured to raise an interrupt when the magnitude of the high-pass filtered acceleration threshold is exceeded. The TRANSIENT_CFG register is used to enable the transient interrupt generation mechanism for the 3 axes (X, Y, Z) of acceleration. There is also an option to bypass the high-pass filter. When the high-pass filter is bypassed, the function behaves similar to the motion detection.

0x1D: TRANSIENT_CFG Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
—	—	—	ELE	ZTEFE	YTEFE	XTEFE	HPF_BYP

Table 39. TRANSIENT_CFG Description

ELE	Transient event flags are latched into the TRANSIENT_SRC register. Reading of the TRANSIENT_SRC register clears the event flag. Default value: 0. 0: Event flag latch disabled; 1: Event flag latch enabled
ZTEFE	Event flag enable on Z transient acceleration greater than transient threshold event. Default value: 0. 0: Event detection disabled; 1: Raise event flag on measured acceleration delta value greater than transient threshold.
YTEFE	Event flag enable on Y transient acceleration greater than transient threshold event. Default value: 0. 0: Event detection disabled; 1: Raise event flag on measured acceleration delta value greater than transient threshold.
XTEFE	Event flag enable on X transient acceleration greater than transient threshold event. Default value: 0. 0: Event detection disabled; 1: Raise event flag on measured acceleration delta value greater than transient threshold.
HPF_BYP	Bypass High-Pass filter Default value: 0. 0: Data to transient acceleration detection block is through HPF 1: Data to transient acceleration detection block is NOT through HPF (similar to motion detection function)

0x1E: TRANSIENT_SRC Register

The Transient Source register provides the status of the enabled axes and the polarity (directional) information. When this register is read it clears the interrupt for the transient detection. When new events arrive while EA = 1, additional *TRANSE bits may get set, and the corresponding *_Trans_Pol flag become updated. However, no *TRANSE bit may get cleared before the TRANSIENT_SRC register is read.

0x1E: TRANSIENT_SRC Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
—	EA	ZTRANSE	Z_Trans_Pol	YTRANSE	Y_Trans_Pol	XTRANSE	X_Trans_Pol

Table 40. TRANSIENT_SRC Description

EA	Event Active Flag. Default value: 0. 0: no event flag has been asserted; 1: one or more event flag has been asserted.
ZTRANSE	Z transient event. Default value: 0. 0: no interrupt, 1: Z Transient acceleration greater than the value of TRANSIENT_THS event has occurred
Z_Trans_Pol	Polarity of Z Transient Event that triggered interrupt. Default value: 0. 0: Z event was Positive g, 1: Z event was Negative g
YTRANSE	Y transient event. Default value: 0. 0: no interrupt, 1: Y Transient acceleration greater than the value of TRANSIENT_THS event has occurred
Y_Trans_Pol	Polarity of Y Transient Event that triggered interrupt. Default value: 0. 0: Y event was Positive g, 1: Y event was Negative g
XTRANSE	X transient event. Default value: 0. 0: no interrupt, 1: X Transient acceleration greater than the value of TRANSIENT_THS event has occurred
X_Trans_Pol	Polarity of X Transient Event that triggered interrupt. Default value: 0. 0: X event was Positive g, 1: X event was Negative g

When the EA bit gets set while ELE = 1, all other status bits get frozen at their current state. By reading the TRANSIENT_SRC register, all bits get cleared.

0x1F: TRANSIENT_THS Register

The Transient Threshold register sets the threshold limit for the detection of the transient acceleration. The value in the TRANSIENT_THS register corresponds to a g value which is compared against the values of High-Pass Filtered Data. If the High-Pass Filtered acceleration value exceeds the threshold limit, an event flag is raised and the interrupt is generated if enabled.

0x1F: TRANSIENT_THS Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBCNTM	THS6	THS5	THS4	THS3	THS2	THS1	THS0

Table 41. TRANSIENT_THS Description

DBCNTM	Debounce counter mode selection. Default value: 0. 0: increments or decrements debounce; 1: increments or clears counter.
THS[6:0]	Transient Threshold: Default value: 000_0000.

The threshold THS[6:0] is a 7-bit unsigned number, 0.063g/LSB. The maximum threshold is 8g. Even if the part is set to full scale at 2g or 4g this function will still operate up to 8g. If the Low Noise bit is set in Register 0x2A, the maximum threshold to be reached is 4g.

0x20: TRANSIENT_COUNT

The TRANSIENT_COUNT sets the minimum number of debounce counts continuously matching the condition where the unsigned value of high-pass filtered data is greater than the user specified value of TRANSIENT_THS.

0x20: TRANSIENT_COUNT Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

Table 42. TRANSIENT_COUNT Description

D[7:0]	Count value. Default value: 0000_0000.
--------	--

The time step for the transient detection debounce counter is set by the value of the system ODR and the Oversampling mode.

Table 43. TRANSIENT_COUNT Relationship with the ODR

ODR (Hz)	Max Time Range (s)				Time Step (ms)			
	Normal	LPLN	HighRes	LP	Normal	LPLN	HighRes	LP
800	0.319	0.319	0.319	0.319	1.25	1.25	1.25	1.25
400	0.638	0.638	0.638	0.638	2.5	2.5	2.5	2.5
200	1.28	1.28	0.638	1.28	5	5	2.5	5
100	2.55	2.55	0.638	2.55	10	10	2.5	10
50	5.1	5.1	0.638	5.1	20	20	2.5	20
12.5	5.1	20.4	0.638	20.4	20	80	2.5	80
6.25	5.1	20.4	0.638	40.8	20	80	2.5	160
1.56	5.1	20.4	0.638	40.8	20	80	2.5	160

6.6 Single, Double and Directional Tap Detection Registers

For more details of how to configure the tap detection and sample code, please refer to Freescale application note, AN4072. The tap detection registers are referred to as “Pulse”.

0x21: PULSE_CFG Pulse Configuration Register

This register configures the event flag for the tap detection for enabling/disabling the detection of a single and double pulse on each of the axes.

0x21: PULSE_CFG Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DPA	ELE	ZDPEFE	ZSPEFE	YDPEFE	YSPEFE	XDPEFE	XSPEFE

Table 44. PULSE_CFG Description

DPA	Double Pulse Abort. Default value: 0. 0: Double Pulse detection is not aborted if the start of a pulse is detected during the time period specified by the PULSE_LTCY register. 1: Setting the DPA bit momentarily suspends the double tap detection if the start of a pulse is detected during the time period specified by the PULSE_LTCY register and the pulse ends before the end of the time period specified by the PULSE_LTCY register.
ELE	Pulse event flags are latched into the PULSE_SRC register. Reading of the PULSE_SRC register clears the event flag. Default value: 0. 0: Event flag latch disabled; 1: Event flag latch enabled
ZDPEFE	Event flag enable on double pulse event on Z-axis. Default value: 0. 0: Event detection disabled; 1: Event detection enabled
ZSPEFE	Event flag enable on single pulse event on Z-axis. Default value: 0. 0: Event detection disabled; 1: Event detection enabled
YDPEFE	Event flag enable on double pulse event on Y-axis. Default value: 0. 0: Event detection disabled; 1: Event detection enabled
YSPEFE	Event flag enable on single pulse event on Y-axis. Default value: 0. 0: Event detection disabled; 1: Event detection enabled
XDPEFE	Event flag enable on double pulse event on X-axis. Default value: 0. 0: Event detection disabled; 1: Event detection enabled
XSPEFE	Event flag enable on single pulse event on X-axis. Default value: 0. 0: Event detection disabled; 1: Event detection enabled

0x22: PULSE_SRC Pulse Source Register

This register indicates a double or single pulse event has occurred and also which direction. The corresponding axis and event must be enabled in Register 0x21 for the event to be seen in the source register.

0x22: PULSE_SRC Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EA	AxZ	AxY	AxX	DPE	PolZ	PolY	PolX

Table 45. PULSE_SRC Description

EA	Event Active Flag. Default value: 0. (0: No interrupt has been generated; 1: One or more interrupt events have been generated)
AxZ	Z-axis event. Default value: 0. (0: No interrupt; 1: Z-axis event has occurred)
AxY	Y-axis event. Default value: 0. (0: No interrupt; 1: Y-axis event has occurred)
AxX	X-axis event. Default value: 0. (0: No interrupt; 1: X-axis event has occurred)
DPE	Double pulse on first event. Default value: 0. (0: Single Pulse Event triggered interrupt; 1: Double Pulse Event triggered interrupt)
PolZ	Pulse polarity of Z-axis Event. Default value: 0. (0: Pulse Event that triggered interrupt was Positive; 1: Pulse Event that triggered interrupt was negative)
PolY	Pulse polarity of Y-axis Event. Default value: 0. (0: Pulse Event that triggered interrupt was Positive; 1: Pulse Event that triggered interrupt was negative)
PolX	Pulse polarity of X-axis Event. Default value: 0. (0: Pulse Event that triggered interrupt was Positive; 1: Pulse Event that triggered interrupt was negative)

When the EA bit gets set while ELE = 1, all status bits (AxZ, AxY, AxZ, DPE, and PolX, PolY, PolZ) are frozen. Reading the PULSE_SRC register clears all bits. Reading the source register will clear the interrupt.

MMA8451Q

0x23 - 0x25: PULSE_THSX, Y, Z Pulse Threshold for X, Y & Z Registers

The pulse threshold can be set separately for the X, Y and Z axes. The PULSE_THSX, PULSE_THSY and PULSE_THSZ registers define the threshold which is used by the system to start the pulse detection procedure.

0x23: PULSE_THSX Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	THSX6	THSX5	THSX4	THSX3	THSX2	THSX1	THSX0

Table 46. PULSE_THSX Description

THSX[6:0]	Pulse Threshold on X-axis. Default value: 000_0000.
-----------	---

0x24: PULSE_THSY Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	THSY6	THSY5	THSY4	THSY3	THSY2	THSY1	THSY0

Table 47. PULSE_THSY Description

THSY[6:0]	Pulse Threshold on Y-axis. Default value: 000_0000.
-----------	---

0x25: PULSE_THSZ Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	THSZ6	THSZ5	THSZ4	THSZ3	THSZ2	THSZ1	THSZ0

Table 48. PULSE_THSZ Description

THSZ[6:0]	Pulse Threshold on Z-axis. Default value: 000_0000.
-----------	---

The threshold values range from 1 to 127 with steps of 0.63g/LSB at a fixed 8g acceleration range, thus the minimum resolution is always fixed at 0.063g/LSB. If the Low Noise bit in Register 0x2A is set then the maximum threshold will be 4g. The PULSE_THSX, PULSE_THSY and PULSE_THSZ registers define the threshold which is used by the system to start the pulse detection procedure. The threshold value is expressed over 7-bits as an unsigned number.

0x26: PULSE_TMLT Pulse Time Window 1 Register

0x26: PULSE_TMLT Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TMLT7	TMLT6	TMLT5	TMLT4	TMLT3	TMLT2	TMLT1	TMLT0

Table 49. PULSE_TMLT Description

TMLT[7:0]	Pulse Time Limit. Default value: 0000_0000.
-----------	---

The bits TMLT7 through TMLT0 define the maximum time interval that can elapse between the start of the acceleration on the selected axis exceeding the specified threshold and the end when the acceleration on the selected axis must go below the specified threshold to be considered a valid pulse.

The minimum time step for the pulse time limit is defined in [Table 50](#) and [Table 51](#). Maximum time for a given ODR and Oversampling mode is the time step pulse multiplied by 255. The time steps available are dependent on the Oversampling mode and whether the Pulse Low-Pass Filter option is enabled or not. The Pulse Low Pass Filter is set in Register 0x0F.

Table 50. Time Step for PULSE Time Limit (Reg 0x0F) Pulse_LPF_EN = 1

ODR (Hz)	Max Time Range (s)				Time Step (ms)			
	Normal	LPLN	HighRes	LP	Normal	LPLN	HighRes	LP
800	0.319	0.319	0.319	0.319	1.25	1.25	1.25	1.25
400	0.638	0.638	0.638	0.638	2.5	2.5	2.5	2.5
200	1.28	1.28	0.638	1.28	5	5	2.5	5
100	2.55	2.55	0.638	2.55	10	10	2.5	10
50	5.1	5.1	0.638	5.1	20	20	2.5	20
12.5	5.1	20.4	0.638	20.4	20	80	2.5	80
6.25	5.1	20.4	0.638	40.8	20	80	2.5	160
1.56	5.1	20.4	0.638	40.8	20	80	2.5	160

Table 51. Time Step for PULSE Time Limit (Reg 0x0F) Pulse_LPF_EN = 0

ODR (Hz)	Max Time Range (s)				Time Step (ms)			
	Normal	LPLN	HighRes	LP	Normal	LPLN	HighRes	LP
800	0.159	0.159	0.159	0.159	0.625	0.625	0.625	0.625
400	0.159	0.159	0.159	0.319	0.625	0.625	0.625	1.25
200	0.319	0.319	0.159	0.638	1.25	1.25	0.625	2.5
100	0.638	0.638	0.159	1.28	2.5	2.5	0.625	5
50	1.28	1.28	0.159	2.55	5	5	0.625	10
12.5	1.28	5.1	0.159	10.2	5	20	0.625	40
6.25	1.28	5.1	0.159	10.2	5	20	0.625	40
1.56	1.28	5.1	0.159	10.2	5	20	0.625	40

0x27: PULSE_LTCY Pulse Latency Timer Register

0x27: PULSE_LTCY Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LTCY7	LTCY6	LTCY5	LTCY4	LTCY3	LTCY2	LTCY1	LTCY0

Table 52. PULSE_LTCY Description

LTCY[7:0]	Latency Time Limit. Default value: 0000_0000
-----------	--

The bits LTCY7 through LTCY0 define the time interval that starts after the first pulse detection. During this time interval, all pulses are ignored. **Note:** This timer must be set for single pulse and for double pulse.

The minimum time step for the pulse latency is defined in Table 53 and Table 54. The maximum time is the time step at the ODR and Oversampling mode multiplied by 255. The timing also changes when the Pulse LPF is enabled or disabled.

Table 53. Time Step for PULSE Latency @ ODR and Power Mode (Reg 0x0F) Pulse_LPF_EN = 1

ODR (Hz)	Max Time Range (s)				Time Step (ms)			
	Normal	LPLN	HighRes	LP	Normal	LPLN	HighRes	LP
800	0.638	0.638	0.638	0.638	2.5	2.5	2.5	2.5
400	1.276	1.276	1.276	1.276	5	5	5	5
200	2.56	2.56	1.276	2.56	10	10	5	10
100	5.1	5.1	1.276	5.1	20	20	5	20
50	10.2	10.2	1.276	10.2	40	40	5	40
12.5	10.2	40.8	1.276	40.8	40	160	5	160
6.25	10.2	40.8	1.276	81.6	40	160	5	320
1.56	10.2	40.8	1.276	81.6	40	160	5	320

Table 54. Time Step for PULSE Latency @ ODR and Power Mode (Reg 0x0F) Pulse_LPF_EN = 0

ODR (Hz)	Max Time Range (s)				Time Step (ms)			
	Normal	LPLN	HighRes	LP	Normal	LPLN	HighRes	LP
800	0.318	0.318	0.318	0.318	1.25	1.25	1.25	1.25
400	0.318	0.318	0.318	0.638	1.25	1.25	1.25	2.5
200	0.638	0.638	0.318	1.276	2.5	2.5	1.25	5
100	1.276	1.276	0.318	2.56	5	5	1.25	10
50	2.56	2.56	0.318	5.1	10	10	1.25	20
12.5	2.56	10.2	0.318	20.4	10	40	1.25	80
6.25	2.56	10.2	0.318	20.4	10	40	1.25	80
1.56	2.56	10.2	0.318	20.4	10	40	1.25	80

0x28: PULSE_WIND Register (Read/Write)

0x28: PULSE_WIND Second Pulse Time Window Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WIND7	WIND6	WIND5	WIND4	WIND3	WIND2	WIND1	WIND0

Table 55. PULSE_WIND Description

WIND[7:0]	Second Pulse Time Window. Default value: 0000_0000.
-----------	---

The bits WIND7 through WIND0 define the maximum interval of time that can elapse after the end of the latency interval in which the start of the second pulse event must be detected provided the device has been configured for double pulse detection. The detected second pulse width must be shorter than the time limit constraints specified by the PULSE_TMLT register, but the end of the double pulse need not finish within the time specified by the PULSE_WIND register.

The minimum time step for the pulse window is defined in [Table 56](#) and [Table 57](#). The maximum time is the time step at the ODR, Oversampling mode and LPF Filter Option multiplied by 255.

Table 56. Time Step for PULSE Detection Window @ ODR and Power Mode (Reg 0x0F) Pulse_LPF_EN = 1

ODR (Hz)	Max Time Range (s)				Time Step (ms)			
	Normal	LPLN	HighRes	LP	Normal	LPLN	HighRes	LP
800	0.638	0.638	0.638	0.638	2.5	2.5	2.5	2.5
400	1.276	1.276	1.276	1.276	5	5	5	5
200	2.56	2.56	1.276	2.56	10	10	5	10
100	5.1	5.1	1.276	5.1	20	20	5	20
50	10.2	10.2	1.276	10.2	40	40	5	40
12.5	10.2	40.8	1.276	40.8	40	160	5	160
6.25	10.2	40.8	1.276	81.6	40	160	5	320
1.56	10.2	40.8	1.276	81.6	40	160	5	320

Table 57. Time Step for PULSE Detection Window @ ODR and Power Mode (Reg 0x0F) Pulse_LPF_EN = 0

ODR (Hz)	Max Time Range (s)				Time Step (ms)			
	Normal	LPLN	HighRes	LP	Normal	LPLN	HighRes	LP
800	0.318	0.318	0.318	0.318	1.25	1.25	1.25	1.25
400	0.318	0.318	0.318	0.638	1.25	1.25	1.25	2.5
200	0.638	0.638	0.318	1.276	2.5	2.5	1.25	5
100	1.276	1.276	0.318	2.56	5	5	1.25	10
50	2.56	2.56	0.318	5.1	10	10	1.25	20
12.5	2.56	10.2	0.318	20.4	10	40	1.25	80
6.25	2.56	10.2	0.318	20.4	10	40	1.25	80
1.56	2.56	10.2	0.318	20.4	10	40	1.25	80

6.7 Auto-WAKE/SLEEP Detection

The ASLP_COUNT register sets the minimum time period of inactivity required to change current ODR value from the value specified in the DR[2:0] register to ASLP_RATE register value, provided the SLPE bit is set to a logic '1' in the CTRL_REG2 register. See Table 59 for functional blocks that may be monitored for inactivity in order to trigger the "return to SLEEP" event.

0x29: ASLP_COUNT Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

Table 58. ASLP_COUNT Description

D[7:0]	Duration value. Default value: 0000_0000.
--------	---

D7-D0 defines the minimum duration time to change current ODR value from DR to ASLP_RATE. Time step and maximum value depend on the ODR chosen as shown in Table 59.

Table 59. ASLP_COUNT Relationship with ODR

Output Data Rate (ODR)	Duration	ODR Time Step	ASLP_COUNT Step
800 Hz	0 to 81s	1.25 ms	320 ms
400 Hz	0 to 81s	2.5 ms	320 ms
200 Hz	0 to 81s	5 ms	320 ms
100 Hz	0 to 81s	10 ms	320 ms
50 Hz	0 to 81s	20 ms	320 ms
12.5 Hz	0 to 81s	80 ms	320 ms
6.25 Hz	0 to 81s	160 ms	320 ms
1.56 Hz	0 to 162s	640 ms	640 ms

Table 60. SLEEP/WAKE Mode Gates and Triggers

Interrupt Source	Event restarts timer and delays Return to SLEEP	Event will WAKE from SLEEP
FIFO_GATE	Yes	No
SRC_TRANS	Yes	Yes
SRC_LNDPRT	Yes	Yes
SRC_PULSE	Yes	Yes
SRC_FF_MT	Yes	Yes
SRC_ASLEEP	No*	No*
SRC_DRDY	No	No

* If the FIFO_GATE bit is set to logic '1', the assertion of the SRC_ASLEEP interrupt does not prevent the system from transitioning to SLEEP or from WAKE mode; instead it prevents the FIFO buffer from accepting new sample data until the host application flushes the FIFO buffer.

In order to wake the device, the desired function or functions must be enabled in CTRL_REG4 and set to WAKE to SLEEP in CTRL_REG3. All enabled functions will still function in SLEEP mode at the SLEEP ODR. Only the functions that have been selected for WAKE from SLEEP will **WAKE** the device.

MMA8451Q has four functions that can be used to keep the sensor from falling asleep; Transient, Orientation, Tap and Motion/Freefall. One or more of these functions can be enabled. In order to WAKE the device, four functions are provided; Transient, Orientation, Tap, and the Motion/Freefall. Note that the FIFO does not WAKE the device. The Auto-WAKE/SLEEP interrupt does not affect the WAKE/SLEEP, nor does the data ready interrupt. The FIFO gate (bit 7) in Register 0x2C, when set, will hold the last data in the FIFO before transitioning to a different ODR. After the buffer is flushed, it will accept new sample data at the current ODR. See Register 0x2C for the WAKE from SLEEP bits.

If the Auto-SLEEP bit is disabled, then the device can only toggle between STANDBY and WAKE mode. If Auto-SLEEP interrupt is enabled, transitioning from ACTIVE mode to Auto-SLEEP mode and vice versa generates an interrupt.

6.8 Control Registers

Note: Except for STANDBY mode selection, the device must be in STANDBY mode to change any of the fields within CTRL_REG1 (0x2A).

0x2A: CTRL_REG1 System Control 1 Register

0x2A: CTRL_REG1 Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ASLP_RATE1	ASLP_RATE0	DR2	DR1	DR0	LNOISE	F_READ	ACTIVE

Table 61. CTRL_REG1 Description

ASLP_RATE[1:0]	Configures the Auto-WAKE sample frequency when the device is in SLEEP Mode. Default value: 00. See Table 62 for more information.
DR[2:0]	Data rate selection. Default value: 000. See Table 63 for more information.
LNOISE	Reduced noise reduced Maximum range mode. Default value: 0. (0: Normal mode; 1: Reduced Noise mode)
F_READ	Fast Read mode: Data format limited to single Byte Default value: 0. (0: Normal mode 1: Fast Read Mode)
ACTIVE	Full Scale selection. Default value: 00. (0: STANDBY mode; 1: ACTIVE mode)

Table 62. SLEEP Mode Rate Description

ASLP_RATE1	ASLP_RATE0	Frequency (Hz)
0	0	50
0	1	12.5
1	0	6.25
1	1	1.56

It is important to note that when the device is Auto-SLEEP mode, the system ODR and the data rate for all the system functional blocks are overridden by the data rate set by the **ASLP_RATE** field.

DR[2:0] bits select the Output Data Rate (ODR) for acceleration samples. The default value is **000 for a data rate of 800 Hz**.

Table 63. System Output Data Rate Selection

DR2	DR1	DR0	ODR	Period
0	0	0	800 Hz	1.25 ms
0	0	1	400 Hz	2.5 ms
0	1	0	200 Hz	5 ms
0	1	1	100 Hz	10 ms
1	0	0	50 Hz	20 ms
1	0	1	12.5 Hz	80 ms
1	1	0	6.25 Hz	160 ms
1	1	1	1.56 Hz	640 ms

ACTIVE bit selects between STANDBY mode and ACTIVE mode. The default value is 0 for STANDBY mode.

Table 64. Full Scale Selection

Active	Mode
0	STANDBY
1	ACTIVE

LNOISE bit selects between normal full dynamic range mode and a high sensitivity, Low Noise mode. In Low Noise mode, the maximum signal that can be measured is $\pm 4g$. **Note:** Any thresholds set above 4g will not be reached.

F_READ bit selects between normal and Fast Read mode. When selected, the auto increment counter will skip over the LSB data bytes. Data read from the FIFO will skip over the LSB data, reducing the acquisition time. Note **F_READ** can only be changed when FMODE = 00. The **F_READ** bit applies for both the output registers and the FIFO.

0x2B: CTRL_REG2 System Control 2 Register

0x2B: CTRL_REG2 Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ST	RST	0	SMODS1	SMODS0	SLPE	MODS1	MODS0

Table 65. CTRL_REG2 Description

ST	Self-Test Enable. Default value: 0. 0: Self-Test disabled; 1: Self-Test enabled
RST	Software Reset. Default value: 0. 0: Device reset disabled; 1: Device reset enabled.
SMODS[1:0]	SLEEP mode power scheme selection. Default value: 00. See Table 66 and Table 67
SLPE	Auto-SLEEP enable. Default value: 0. 0: Auto-SLEEP is not enabled; 1: Auto-SLEEP is enabled.
MODS[1:0]	ACTIVE mode power scheme selection. Default value: 00. See Table 66 and Table 67

ST bit activates the self-test function. When ST is set, X, Y, and Z outputs will shift. **RST** bit is used to activate the software reset. The reset mechanism can be enabled in STANDBY and ACTIVE mode.

When the reset bit is enabled, all registers are reset and are loaded with default values. Writing '1' to the RST bit immediately resets the device, no matter whether it is in ACTIVE/WAKE, ACTIVE/SLEEP, or STANDBY mode.

The I²C communication system is reset to avoid accidental corrupted data access.

At the end of the boot process, the RST bit is deasserted to 0. Reading this bit will return a value of zero.

The **(S)MODS[1:0]** bits select which Oversampling mode is to be used shown in [Table 66](#). The Oversampling modes are available in both WAKE Mode MOD[1:0] and also in the SLEEP Mode SMOD[1:0].

Table 66. MODS Oversampling Modes

(S)MODS1	(S)MODS0	Power Mode
0	0	Normal
0	1	Low Noise Low Power
1	0	High Resolution
1	1	Low Power

Table 67. MODS Oversampling Modes Current Consumption and Averaging Values at each ODR

Mode	Normal (00)		Low Noise Low Power (01)		High Resolution (10)		Low Power (11)	
	Current μ A	OS Ratio	Current μ A	OS Ratio	Current μ A	OS Ratio	Current μ A	OS Ratio
1.56 Hz	24	128	8	32	165	1024	6	16
6.25 Hz	24	32	8	8	165	256	6	4
12.5 Hz	24	16	8	4	165	128	6	2
50 Hz	24	4	24	4	165	32	14	2
100 Hz	44	4	44	4	165	16	24	2
200 Hz	85	4	85	4	165	8	44	2
400 Hz	165	4	165	4	165	4	85	2
800 Hz	165	2	165	2	165	2	165	2

0x2C: CTRL_REG3 Interrupt Control Register

0x2C: CTRL_REG3 Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FIFO_GATE	WAKE_TRANS	WAKE_LNDPRT	WAKE_PULSE	WAKE_FF_MT	—	IPOL	PP_OD

Table 68. CTRL_REG3 Description

FIFO_GATE	0: FIFO gate is bypassed. FIFO is flushed upon the system mode transitioning from WAKE to SLEEP mode or from SLEEP to WAKE mode. Default value: 0. 1: The FIFO input buffer is blocked when transitioning from WAKE to SLEEP mode or from SLEEP to WAKE mode until the FIFO is flushed. Although the system transitions from WAKE to SLEEP or from SLEEP to WAKE the contents of the FIFO buffer are preserved, new data samples are ignored until the FIFO is emptied by the host application. If the FIFO_GATE bit is set to logic '1' and the FIFO buffer is not emptied before the arrival of the next sample, then the FGERR bit in the SYS_MOD register (0x0B) will be asserted. The FGERR bit remains asserted as long as the FIFO buffer remains un-emptied. Emptying the FIFO buffer clears the FGERR bit in the SYS_MOD register.
WAKE_TRANS	0: Transient function is bypassed in SLEEP mode. Default value: 0. 1: Transient function interrupt can wake up system
WAKE_LNDPRT	0: Orientation function is bypassed in SLEEP mode. Default value: 0. 1: Orientation function interrupt can wake up system
WAKE_PULSE	0: Pulse function is bypassed in SLEEP mode. Default value: 0. 1: Pulse function interrupt can wake up system
WAKE_FF_MT	0: Freefall/Motion function is bypassed in SLEEP mode. Default value: 0. 1: Freefall/Motion function interrupt can wake up
IPOL	Interrupt polarity ACTIVE high, or ACTIVE low. Default value: 0. 0: ACTIVE low; 1: ACTIVE high
PP_OD	Push-Pull/Open Drain selection on interrupt pad. Default value: 0. 0: Push-Pull; 1: Open Drain

IPOL bit selects the polarity of the interrupt signal. When IPOL is '0' (default value) any interrupt event will signaled with a logical 0.

PP_OD bit configures the interrupt pin to Push-Pull or in Open Drain mode. The default value is 0 which corresponds to Push-Pull mode. The Open Drain configuration can be used for connecting multiple interrupt signals on the same interrupt line.

0x2D: CTRL_REG4 Register (Read/Write)

0x2D: CTRL_REG4 Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INT_EN_ASLP	INT_EN_FIFO	INT_EN_TRANS	INT_EN_LNDPRT	INT_EN_PULSE	INT_EN_FF_MT	—	INT_EN_DRDY

Table 69. Interrupt Enable Register Description

Interrupt Enable	Description
INT_EN_ASLP	Interrupt Enable. Default value: 0. 0: Auto-SLEEP/WAKE interrupt disabled; 1: Auto-SLEEP/WAKE interrupt enabled.
INT_EN_FIFO	Interrupt Enable. Default value: 0. 0: FIFO interrupt disabled; 1: FIFO interrupt enabled.
INT_EN_TRANS	Interrupt Enable. Default value: 0. 0: Transient interrupt disabled; 1: Transient interrupt enabled.
INT_EN_LNDPRT	Interrupt Enable. Default value: 0. 0: Orientation (Landscape/Portrait) interrupt disabled. 1: Orientation (Landscape/Portrait) interrupt enabled.
INT_EN_PULSE	Interrupt Enable. Default value: 0. 0: Pulse Detection interrupt disabled; 1: Pulse Detection interrupt enabled
INT_EN_FF_MT	Interrupt Enable. Default value: 0. 0: Freefall/Motion interrupt disabled; 1: Freefall/Motion interrupt enabled
INT_EN_DRDY	Interrupt Enable. Default value: 0. 0: Data Ready interrupt disabled; 1: Data Ready interrupt enabled

The corresponding functional block interrupt enable bit allows the functional block to route its event detection flags to the system's interrupt controller. The interrupt controller routes the enabled functional block interrupt to the INT1 or INT2 pin.

0x2E: CTRL_REG5 Register (Read/Write)

0x2E: CTRL_REG5 Interrupt Configuration Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INT_CFG_ASLP	INT_CFG_FIFO	INT_CFG_TRANS	INT_CFG_LNDPRT	INT_CFG_PULSE	INT_CFG_FF_MT	—	INT_CFG_DRDY

Table 70. Interrupt Configuration Register Description

Interrupt Configuration	Description
INT_CFG_ASLP	INT1/INT2 Configuration. Default value: 0. 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin
INT_CFG_FIFO	INT1/INT2 Configuration. Default value: 0 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin
INT_CFG_TRANS	INT1/INT2 Configuration. Default value: 0. 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin
INT_CFG_LNDPRT	INT1/INT2 Configuration. Default value: 0. 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin
INT_CFG_PULSE	INT1/INT2 Configuration. Default value: 0. 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin
INT_CFG_FF_MT	INT1/INT2 Configuration. Default value: 0. 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin
INT_CFG_DRDY	INT1/INT2 Configuration. Default value: 0. 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin

The system's interrupt controller shown in [Figure 10](#) uses the corresponding bit field in the CTRL_REG5 register to determine the routing table for the INT1 and INT2 interrupt pins. If the bit value is logic '0', the functional block's interrupt is routed to INT2, and if the bit value is logic '1', then the interrupt is routed to INT1. One or more functions can assert an interrupt pin; therefore a host application responding to an interrupt should read the INT_SOURCE (0x0C) register to determine the appropriate sources of the interrupt.

6.9 User Offset Correction Registers

For more information on how to calibrate the 0g offset, refer to application note AN4069. The 2's complement offset correction registers values are used to realign the Zero-g position of the X, Y, and Z-axis after device board mount. The resolution of the offset registers is 2 mg per LSB. The 2's complement 8-bit value would result in an offset compensation range ± 256 mg.

0x2F: OFF_X Offset Correction X Register

0x2F: OFF_X Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

Table 71. OFF_X Description

D[7:0]	X-axis offset value. Default value: 0000_0000.
--------	--

0x30: OFF_Y Offset Correction Y Register

0x30: OFF_Y Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

Table 72. OFF_Y Description

D[7:0]	Y-axis offset value. Default value: 0000_0000.
--------	--

0x31: OFF_Z Offset Correction Z Register

0x31: OFF_Z Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

Table 73. OFF_Z Description

D[7:0]	Z-axis offset value. Default value: 0000_0000.
--------	--

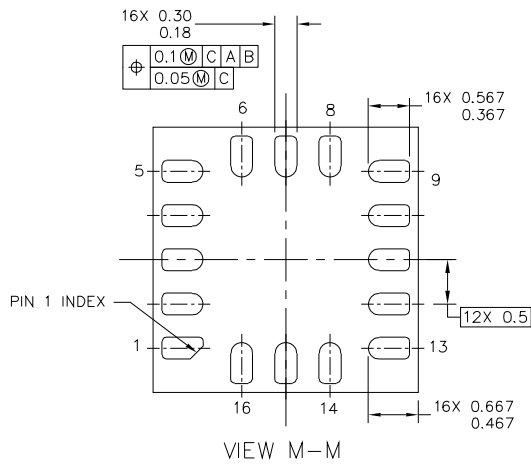
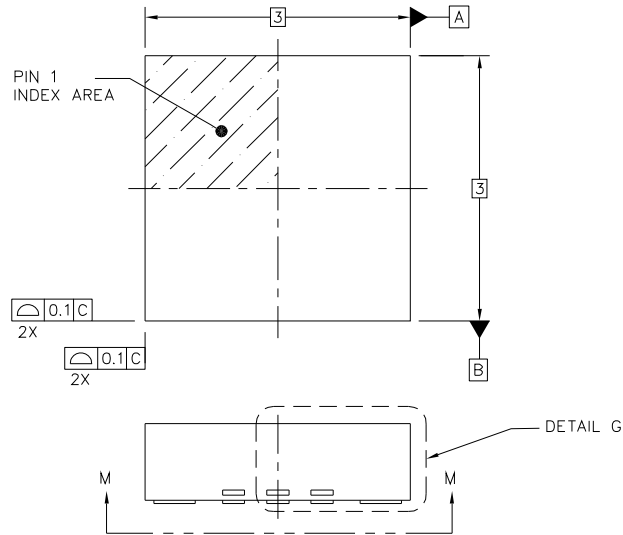
Table 74. MMA8451Q Register Map

Reg	Name	Definition	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00	STATUS/F_STATUS	Data Status R	ZYXOW	ZOW	YOW	XOW	ZYXDR	ZDR	YDR	XDR
01	OUT_X_MSB	14 bit X Data R	XD13	XD12	XD11	XD10	XD9	XD8	XD7	XD6
02	OUT_X_LSB	14 bit X Data R	XD5	XD4	XD3	XD2	XD1	XD0	0	0
03	OUT_Y_MSB	14 bit Y Data R	YD13	YD12	YD11	YD10	YD9	YD8	YD7	YD6
04	OUT_Y_LSB	14 bit Y Data R	YD5	YD4	YD3	YD2	YD1	YD0	0	0
05	OUT_Z_MSB	14 bit Z Data R	ZD13	ZD12	ZD11	ZD10	ZD9	ZD8	ZD7	ZD6
06	OUT_Z_LSB	14 bit Z Data R	ZD5	ZD4	ZD3	ZD2	ZD1	ZD0	0	0
09	F_SETUP	FIFO Setup R/W	F_MODE1	F_MODE0	F_WMRK5	F_WMRK4	F_WMRK3	F_WMRK2	F_WMRK1	F_WMRK0
0A	TRIG_CFG	FIFO Triggers R/W	—	—	Trig_TRANS	Trig_LNDPRT	Trig_PULSE	Trig_FF_MT	—	—
0B	SYSMOD	System Mode R	FGERR	FGT_4	FGT_3	FGT_2	FGT_1	FGT_0	SYSMOD1	SYSMOD0
0C	INT_SOURCE	Interrupt Status R	SRC_ASLP	SRC_FIFO	SRC_TRANS	SRC_LNDPRT	SRC_PULSE	SRC_FF_MT	—	SRC_DRDY
0D	WHO_AM_I	ID Register R	0	0	0	1	1	0	1	0
0E	XYZ_DATA_CFG	Data Config R/W	—	—	—	HPF_Out	—	—	FS1	FS0
0F	HP_FILTER_CUTOFF	HP Filter Setting R/W	—	—	Pulse_HPF_BYP	Pulse_LPF_EN	—	—	SEL1	SEL0
10	PL_STATUS	PL Status R	NEWLP	LO	—	—	—	LAPO[1]	LAPO[0]	BAFRO
11	PL_CFG	PL Configuration R/W	DBCNTM	PL_EN	—	—	—	—	—	—
12	PL_COUNT	PL DEBOUNCE R/W	DBNCE[7]	DBNCE[6]	DBNCE[5]	DBNCE[4]	DBNCE[3]	DBNCE[2]	DBNCE[1]	DBNCE[0]
13	PL_BF_ZCOMP	PL Back/Front Z Comp R/W	BKFR[1]	BKFR[0]	—	—	—	ZLOCK[2]	ZLOCK[1]	ZLOCK[0]
14	P_L_THS_REG	PL THRESHOLD R/W	P_L_THS[4]	P_L_THS[3]	P_L_THS[2]	P_L_THS[1]	P_L_THS[0]	HYS[2]	HYS[1]	HYS[0]
15	FF_MT_CFG	Freefall/Motion Config R/W	ELE	OAE	ZEFE	YEFE	XEFE	—	—	—
16	FF_MT_SRC	Freefall/Motion Source R	EA	—	ZHE	ZHP	YHE	YHP	XHE	XHP
17	FF_MT_THS	Freefall/Motion Threshold R/W	DBCNTM	THS6	THS5	THS4	THS3	THS2	THS1	THS0
18	FF_MT_COUNT	Freefall/Motion Debounce R/W	D7	D6	D5	D4	D3	D2	D1	D0
1D	TRANSIENT_CFG	Transient Config R/W	—	—	—	ELE	ZTEFE	YTEFE	XTEFE	HPF_BYP
1E	TRANSIENT_SRC	Transient Source R	—	EA	ZTRANSE	Z_Trans_Pol	YTRANSE	Y_Trans_Pol	XTRANSE	X_Trans_Pol
1F	TRANSIENT_THS	Transient Threshold R/W	DBCNTM	THS6	THS5	THS4	THS3	THS2	THS1	THS0
20	TRANSIENT_COUNT	Transient Debounce R/W	D7	D6	D5	D4	D3	D2	D1	D0
21	PULSE_CFG	Pulse Config R/W	DPA	ELE	ZDPEFE	ZSPEFE	YDPEFE	YSPEFE	XDPEFE	XSPEFE
22	PULSE_SRC	Pulse Source R	EA	AxZ	AxY	AxX	DPE	PoL_Z	PoL_Y	PoL_X
23	PULSE_THSX	Pulse X Threshold R/W	—	THSX6	THSX5	THSX4	THSX3	THSX2	THSX1	THSX0
24	PULSE_THSY	Pulse Y Threshold R/W	—	THSY6	THSY5	THSY4	THSY3	THSY2	THSY1	THSY0
25	PULSE_THSZ	Pulse Z Threshold R/W	—	THSZ6	THSZ5	THSZ4	THSZ3	THSZ2	THSZ1	THSZ0
26	PULSE_TMLT	Pulse First Timer R/W	TMLT7	TMLT6	TMLT5	TMLT4	TMLT3	TMLT2	TMLT1	TMLT0
27	PULSE_LTCY	Pulse Latency R/W	LTCY7	LTCY6	LTCY5	LTCY4	LTCY3	LTCY2	LTCY1	LTCY0
28	PULSE_WIND	Pulse 2nd Window R/W	WIND7	WIND6	WIND5	WIND4	WIND3	WIND2	WIND1	WIND0
29	ASLP_COUNT	Auto-SLEEP Counter R/W	D7	D6	D5	D4	D3	D2	D1	D0
2A	CTRL_REG1	Control Reg1 R/W	ASLP_RATE1	ASLP_RATE0	DR2	DR1	DR0	LNOISE	F_READ	ACTIVE
2B	CTRL_REG2	Control Reg2 R/W	ST	RST	—	SMODS1	SMODS0	SLPE	MODS1	MODS0
2C	CTRL_REG3	Control Reg3 (WAKE Interrupts from SLEEP) R/W	FIFO_GATE	WAKE_TRANS	WAKE_LNDPRT	WAKE_PULSE	WAKE_FF_MT	—	IPOL	PP_OD
2D	CTRL_REG4	Control Reg4 (Interrupt Enable Map) R/W	INT_EN_ASLP	INT_EN_FIFO	INT_EN_TRANS	INT_EN_LNDPRT	INT_EN_PULSE	INT_EN_FF_MT	—	INT_EN_DRDY
2E	CTRL_REG5	Control Reg5 (Interrupt Configuration) R/W	INT_CFG_ASLP	INT_CFG_FIFO	INT_CFG_TRANS	INT_CFG_LNDPRT	INT_CFG_PULSE	INT_CFG_FF_MT	—	INT_CFG_DRDY
2F	OFF_X	X 8-bit offset R/W	D7	D6	D5	D4	D3	D2	D1	D0
30	OFF_Y	Y 8-bit offset R/W	D7	D6	D5	D4	D3	D2	D1	D0
31	OFF_Z	Z 8-bit offset R/W	D7	D6	D5	D4	D3	D2	D1	D0

Table 75. Accelerometer Output Data

14-bit Data	Range ±2g (0.25 mg)	Range ±4g (0.5 mg)	Range ±8g (1.0 mg)
01 1111 1111 1111	1.99975g	+3.9995g	+7.999g
01 1111 1111 1110	1.99950g	+3.9990g	+7.998g
...
00 0000 0000 0001	0.00025g	+0.0005g	+0.001g
00 0000 0000 0000	0.00000g	0.00000g	0.000g
11 1111 1111 1111	-0.00025g	-0.0005g	-0.001g
...
10 0000 0000 0001	-1.99975g	-3.9995g	-7.999g
10 0000 0000 0000	-2.00000g	-4.0000g	-8.000g
8-bit Data	Range ±2g (15.6 mg)	Range ±4g (31.25 mg)	Range ±8g (62.5 mg)
0111 1111	1.9844g	+3.9688g	+7.9375g
0111 1110	1.9688g	+3.9375g	+7.8750g
...
0000 0001	+0.0156g	+0.0313g	+0.0625g
0000 0000	0.000g	0.0000g	0.0000g
1111 1111	-0.0156g	-0.0313g	-0.0625g
...
1000 0001	-1.9844g	-3.9688g	-7.9375g
1000 0000	-2.0000g	-4.0000g	-8.0000g

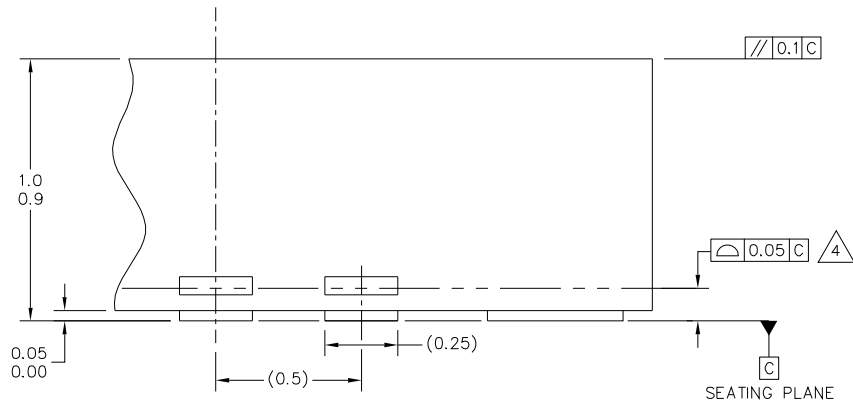
PACKAGE DIMENSIONS



© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: QUAD FLAT NO LEAD COL PACKAGE (QFN-COL) 16 TERMINAL, 0.5 PITCH (3 X 3 X 1.0)	DOCUMENT NO: 98ASA00063D	REV: A
	CASE NUMBER: 2077-02	20 OCT 2011
	STANDARD: NON JEDEC	

CASE 2077-02
ISSUE A
16-LEAD QFN

PACKAGE DIMENSIONS




© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE	
TITLE: QUAD FLAT NO LEAD COL PACKAGE (QFN-COL) 16 TERMINAL, 0.5 PITCH (3 X 3 X 1.0)	DOCUMENT NO: 98ASA00063D	REV: A	
	CASE NUMBER: 2077-02	20 OCT 2011	
	STANDARD: NON JEDEC		

**CASE 2077-02
ISSUE A
16-LEAD QFN**

MMA8451Q

PACKAGE DIMENSIONS

NOTES:

1. ALL DIMENSIONS ARE IN MILLIMETERS.
2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5M-1994.
3. THIS IS NON JEDEC REGISTERED PACKAGE.
4.  COPLANARITY APPLIES TO ALL LEADS.
5. MIN. METAL GAP SHOULD BE 0.2MM.

<small>© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.</small>	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE	
TITLE: QUAD FLAT NO LEAD COL PACKAGE (QFN-COL) 16 TERMINAL, 0.5 PITCH (3 X 3 X 1.0)	DOCUMENT NO: 98ASA00063D	REV: A	
	CASE NUMBER: 2077-02	20 OCT 2011	
	STANDARD: NON JEDEC		

**CASE 2077-02
ISSUE A
16-LEAD QFN**

Table 76. Revision History

Revision number	Revision date	Description of changes
7	03/2012	<ul style="list-style-type: none">• Table 2. Updated Typ values for Sensitivity Accuracy from 2.5% to 2.64%; Zero-g Level Offset Accuracy from ± 20 mg to ± 17 mg and Zero-g Level Offset Accuracy Post Board Mount from ± 30 mg to ± 20 mg.• Table 4. Updated Min value from 50 μs to 0.05 μs and added Max value of 0.9 μs• Added Table 8. Features of the MMA845xQ devices.• Removed FIFO paragraph at the end of Section 6.1.• Updated Note preceding Table 32 from "THS + HYS > 0 and THS + HYS < 49..." to "The condition THS + HYS > 0 and THS + HYS < 32 must be met in order for Landscape/Portrait detection to work properly...."• Updated Case outline with current version.
7.1	05/2012	<ul style="list-style-type: none">• Updated Figure 5 to correspond to table.
8	02/2013	<ul style="list-style-type: none">• Replaced Section 2.3 I²C interface characteristics, including Table 4 and Figure 5.
8.1	10/2013	<ul style="list-style-type: none">• Table 3: Updated Parameter and Test Condition column definitions for "Time from VDDIO on...", "Turn-on (STANDBY)" and "Turn-on time (Power Down to STANDBY)" rows.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

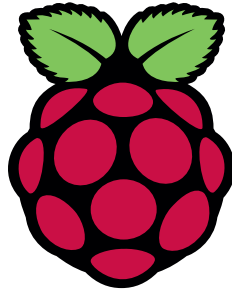
Freescale, the Freescale logo, and the Energy Efficient Solutions logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Xtrinsic is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2013 Freescale Semiconductor, Inc.

Document Number: MMA8451Q
Rev. 8.1
10/2013



DATASHEET



Raspberry Pi Compute Module 3+ **Raspberry Pi Compute Module 3+ Lite**

Release 1, January 2019

Copyright 2019 Raspberry Pi (Trading) Ltd. All rights reserved.



Table 1: Release History

Release	Date	Description
1	28/01/2019	First release

The latest release of this document can be found at <https://www.raspberrypi.org>



Contents

1	Introduction	5
2	Features	6
2.1	Hardware	6
2.2	Peripherals	6
2.3	Software	6
3	Block Diagram	7
4	Mechanical Specification	8
5	Pin Assignments	9
6	Electrical Specification	11
7	Power Supplies	13
7.1	Supply Sequencing	14
7.2	Power Requirements	14
8	Booting	14
9	Peripherals	15
9.1	GPIO	15
9.1.1	GPIO Alternate Functions	16
9.1.2	Secondary Memory Interface (SMI)	17
9.1.3	Display Parallel Interface (DPI)	17
9.1.4	SD/SDIO Interface	18
9.2	CSI (MIPI Serial Camera)	18
9.3	DSI (MIPI Serial Display)	18
9.4	USB	18
9.5	HDMI	18
9.6	Composite (TV Out)	19
10	Thermals	19
10.1	Temperature Range	19
11	Availability	19
12	Support	19



List of Figures

1	CM3+ Block Diagram	7
2	CM3+ Mechanical Dimensions	8
3	Digital IO Characteristics	13



List of Tables

1	Release History	1
2	Compute Module 3+ SODIMM Connector Pinout	9
3	Pin Functions	10
4	Absolute Maximum Ratings	11
5	DC Characteristics	12
6	Digital I/O Pin AC Characteristics	12
7	Power Supply Operating Ranges	13
8	Mimimum Power Supply Requirements	14
9	GPIO Bank0 Alternate Functions	16
10	GPIO Bank1 Alternate Functions	17



1 Introduction

The Raspberry Pi Compute Module 3+ (CM3+) is a range of DDR2-SODIMM-mechanically-compatible System on Modules (SoMs) containing processor, memory, eMMC Flash (on non-Lite variants) and supporting power circuitry. These modules allow a designer to leverage the Raspberry Pi hardware and software stack in their own custom systems and form factors. In addition these modules have extra IO interfaces over and above what is available on the Raspberry Pi model A/B boards, opening up more options for the designer.

The CM3+ contains a BCM2837B0 processor (as used on the Raspberry Pi 3B+), 1Gbyte LPDDR2 RAM and eMMC Flash. The CM3+ is currently available in 4 variants, CM3+/8GB, CM3+/16GB, CM3+/32GB and CM3+ Lite, which have 8, 16 and 32 Gigabytes of eMMC Flash, or no eMMC Flash, respectively.

The CM3+ Lite product is the same as CM3+ except the eMMC Flash is not fitted, and the SD/eMMC interface pins are available for the user to connect their own SD/eMMC device.

Note that the CM3+ is electrically identical and, with the exception of higher CPU z-height, physically identical to the legacy CM3 products.

CM3+ modules require a software/firmware image dated November 2018 or newer to function correctly.



2 Features

2.1 Hardware

- Low cost
- Low power
- High availability
- High reliability
 - Tested over millions of Raspberry Pis Produced to date
 - Module IO pins have 15 micro-inch hard gold plating over 2.5 micron Nickel

2.2 Peripherals

- 48x GPIO
- 2x I2C
- 2x SPI
- 2x UART
- 2x SD/SDIO
- 1x HDMI 1.3a
- 1x USB2 HOST/OTG
- 1x DPI (Parallel RGB Display)
- 1x NAND interface (SMI)
- 1x 4-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 2-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 4-lane DSI Display Interface (up to 1Gbps per lane)
- 1x 2-lane DSI Display Interface (up to 1Gbps per lane)

2.3 Software

- ARMv8 Instruction Set
- Mature and stable Linux software stack
 - Latest Linux Kernel support
 - Many drivers upstreamed
 - Stable and well supported userland
 - Full availability of GPU functions using standard APIs



3 Block Diagram

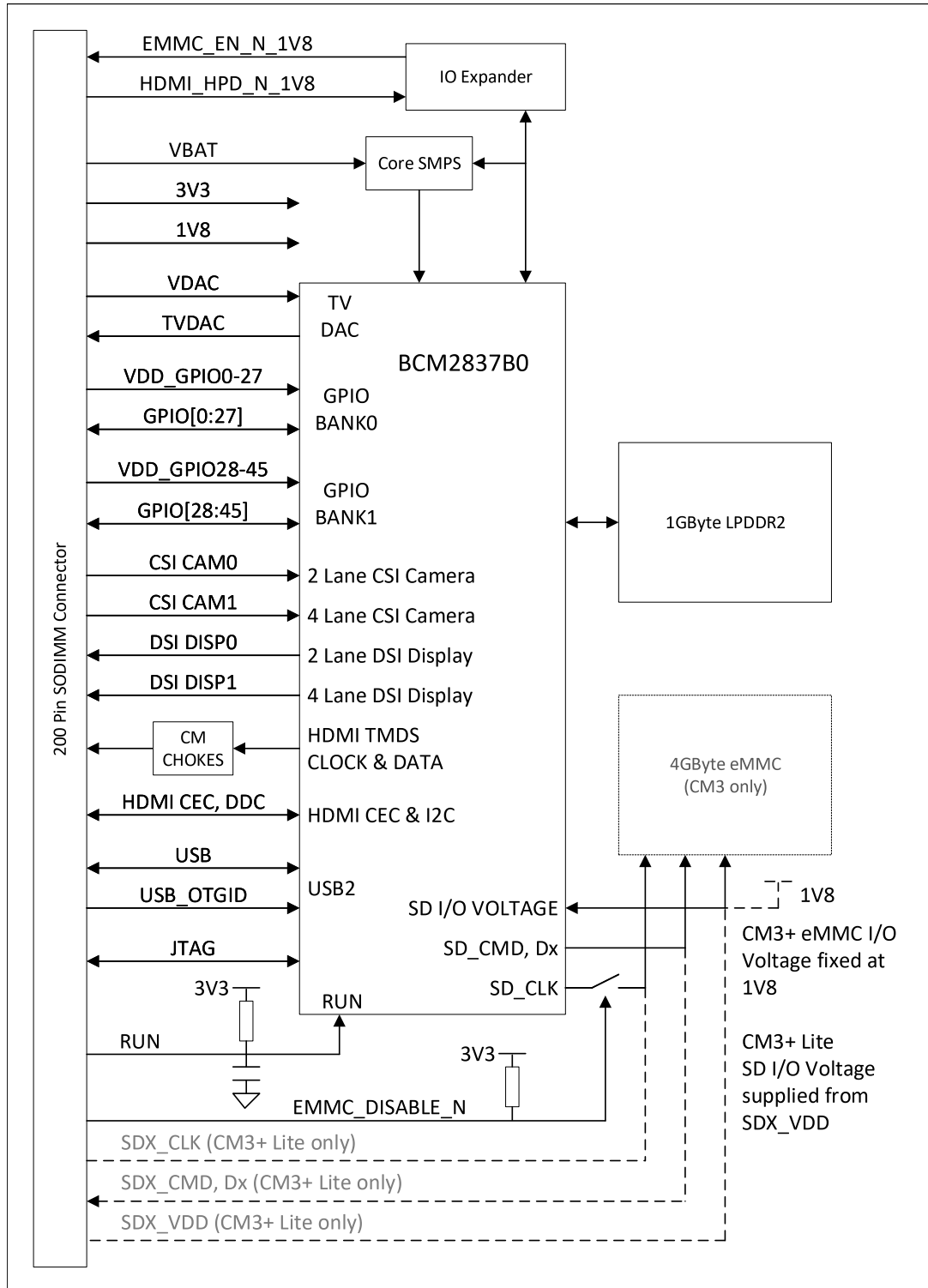


Figure 1: CM3+ Block Diagram



4 Mechanical Specification

The CM3+ modules conform to JEDEC MO-224 mechanical specification for 200 pin DDR2 (1.8V) SODIMM modules and therefore should work with the many DDR2 SODIMM sockets available on the market. **(Please note that the pinout of the Compute Module is not the same as a DDR2 SODIMM module; they are not electrically compatible.)**

The SODIMM form factor was chosen as a way to provide the 200 pin connections using a standard, readily available and low cost connector compatible with low cost PCB manufacture.

The maximum component height on the underside of the Compute Module is 1.2mm.

The maximum component height on the top side of the Compute Module is 2.5mm.

The Compute Module PCB thickness is 1.0mm +/- 0.1mm.

Note that the location and arrangement of components on the Compute Module may change slightly over time due to revisions for cost and manufacturing considerations; however, maximum component heights and PCB thickness will be kept as specified.

Figure 2 gives the CM3+ mechanical dimensions.

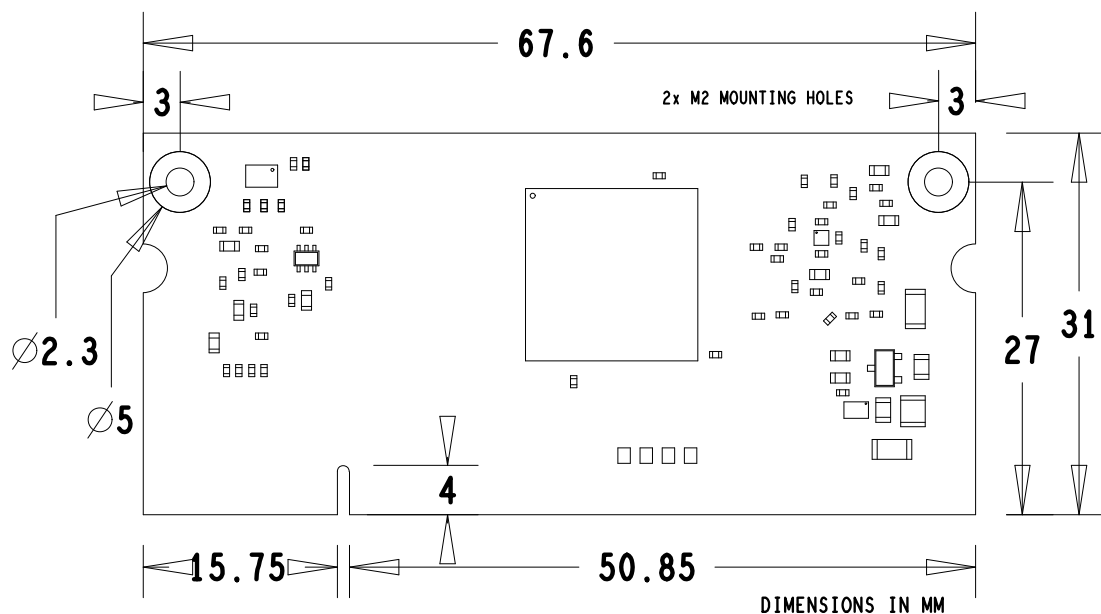


Figure 2: CM3+ Mechanical Dimensions



5 Pin Assignments

CM3+	CM3+ Lite	PIN	PIN	CM3+	CM3+ Lite
GND		1	2	EMMC_DISABLE_N	
GPIO0		3	4	NC	SDX_VDD
GPIO1		5	6	NC	SDX_VDD
GND		7	8	GND	
GPIO2		9	10	NC	SDX_CLK
GPIO3		11	12	NC	SDX_CMD
GND		13	14	GND	
GPIO4		15	16	NC	SDX_D0
GPIO5		17	18	NC	SDX_D1
GND		19	20	GND	
GPIO6		21	22	NC	SDX_D2
GPIO7		23	24	NC	SDX_D3
GND		25	26	GND	
GPIO8		27	28	GPIO28	
GPIO9		29	30	GPIO29	
GND		31	32	GND	
GPIO10		33	34	GPIO30	
GPIO11		35	36	GPIO31	
GND		37	38	GND	
GPIO0-27_VDD		39	40	GPIO0-27_VDD	
KEY					
GPIO28-45_VDD		41	42	GPIO28-45_VDD	
GND		43	44	GND	
GPIO12		45	46	GPIO32	
GPIO13		47	48	GPIO33	
GND		49	50	GND	
GPIO14		51	52	GPIO34	
GPIO15		53	54	GPIO35	
GND		55	56	GND	
GPIO16		57	58	GPIO36	
GPIO17		59	60	GPIO37	
GND		61	62	GND	
GPIO18		63	64	GPIO38	
GPIO19		65	66	GPIO39	
GND		67	68	GND	
GPIO20		69	70	GPIO40	
GPIO21		71	72	GPIO41	
GND		73	74	GND	
GPIO22		75	76	GPIO42	
GPIO23		77	78	GPIO43	
GND		79	80	GND	
GPIO24		81	82	GPIO44	
GPIO25		83	84	GPIO45	
GND		85	86	GND	
GPIO26		87	88	HDMI_HPD_N_1V8	
GPIO27		89	90	EMMC_EN_N_1V8	
GND		91	92	GND	
DSIO_DN1		93	94	DSI1_DP0	
DSIO_DP1		95	96	DSI1_DN0	
GND		97	98	GND	
DSIO_DN0		99	100	DSI1_CP	
DSIO_DP0		101	102	DSI1_CN	
GND		103	104	GND	
DSIO_CN		105	106	DSI1_DP3	
DSIO_CP		107	108	DSI1_DN3	
GND		109	110	GND	
HDMI_CLK_N		111	112	DSI1_DP2	
HDMI_CLK_P		113	114	DSI1_DN2	
GND		115	116	GND	
HDMI_D0_N		117	118	DSI1_DP1	
HDMI_D0_P		119	120	DSI1_DN1	
GND		121	122	GND	
HDMI_D1_N		123	124	NC	
HDMI_D1_P		125	126	NC	
GND		127	128	NC	
HDMI_D2_N		129	130	NC	
HDMI_D2_P		131	132	NC	
GND		133	134	GND	
CAM1_DP3		135	136	CAM0_DP0	
CAM1_DN3		137	138	CAM0_DN0	
GND		139	140	GND	
CAM1_DP2		141	142	CAM0_CP	
CAM1_DN2		143	144	CAM0_CN	
GND		145	146	GND	
CAM1_CP		147	148	CAM0_DP1	
CAM1_CN		149	150	CAM0_DN1	
GND		151	152	GND	
CAM1_DP1		153	154	NC	
CAM1_DN1		155	156	NC	
GND		157	158	NC	
CAM1_DP0		159	160	NC	
CAM1_DN0		161	162	NC	
GND		163	164	GND	
USB_DP		165	166	TVDAC	
USB_DM		167	168	USB_OTGID	
GND		169	170	GND	
HDMI_CEC		171	172	VC_TRST_N	
HDMI_SDA		173	174	VC_TDI	
HDMI_SCL		175	176	VC_TMS	
RUN		177	178	VC_TDO	
DD_CORE (DO NOT CONNECT)		179	180	VC_TCK	
GND		181	182	GND	
1V8		183	184	1V8	
1V8		185	186	1V8	
GND		187	188	GND	
VDAC		189	190	VDAC	
3V3		191	192	3V3	
3V3		193	194	3V3	
GND		195	196	GND	
VBAT		197	198	VBAT	
VBAT		199	200	VBAT	

Table 2: Compute Module 3+ SODIMM Connector Pinout

Table 2 gives the Compute Module 3+ pinout and Table 3 gives the pin functions.



Pin Name	DIR	Voltage Ref	PDN ^a State	If Unused	Description/Notes
<i>RUN and Boot Control (see text for usage guide)</i>					
RUN	I	3V3 ^b	Pull High	Leave open	Has internal 10k pull up
EMMC_DISABLE_N	I	3V3 ^b	Pull High	Leave open	Has internal 10k pull up
EMMC_EN_N_1V8	O	1V8	Pull High	Leave open	Has internal 2k2 pull up
<i>GPIO</i>					
GPIO[27:0]	I/O	GPIO0-27_VDD	Pull or Hi-Z ^c	Leave open	GPIO Bank 0
GPIO[45:28]	I/O	GPIO28-45_VDD	Pull or Hi-Z ^c	Leave open	GPIO Bank 1
<i>Primary SD Interface^{d,e}</i>					
SDX_CLK	O	SDX_VDD	Pull High	Leave open	Primary SD interface CLK
SDX_CMD	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface CMD
SDX_Dx	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface DATA
<i>USB Interface</i>					
USB_Dx	I/O	-	Z	Leave open	Serial interface
USB_OTGID	I	3V3		Tie to GND	OTG pin detect
<i>HDMI Interface</i>					
HDMI_SCL	I/O	3V3 ^b	Z ^f	Leave open	DDC Clock (5.5V tolerant)
HDMI_SDA	I/O	3V3 ^b	Z ^f	Leave open	DDC Data (5.5V tolerant)
HDMI_CEC	I/O	3V3	Z	Leave open	CEC (has internal 27k pull up)
HDMI_CLKx	O	-	Z	Leave open	HDMI serial clock
HDMI_Dx	O	-	Z	Leave open	HDMI serial data
HDMI_HPD_N_1V8	I	1V8	Pull High	Leave open	HDMI hotplug detect
<i>CAM0 (CSI0) 2-lane Interface</i>					
CAM0_Cx	I	-	Z	Leave open	Serial clock
CAM0_Dx	I	-	Z	Leave open	Serial data
<i>CAM1 (CSI1) 4-lane Interface</i>					
CAM1_Cx	I	-	Z	Leave open	Serial clock
CAM1_Dx	I	-	Z	Leave open	Serial data
<i>DSI0 (Display 0) 2-lane Interface</i>					
DSI0_Cx	O	-	Z	Leave open	Serial clock
DSI0_Dx	O	-	Z	Leave open	Serial data
<i>DSI1 (Display 1) 4-lane Interface</i>					
DSI1_Cx	O	-	Z	Leave open	Serial clock
DSI1_Dx	O	-	Z	Leave open	Serial data
<i>TV Out</i>					
TVDAC	O	-	Z	Leave open	Composite video DAC output
<i>JTAG Interface</i>					
TMS	I	3V3	Z	Leave open	Has internal 50k pull up
TRST_N	I	3V3	Z	Leave open	Has internal 50k pull up
TCK	I	3V3	Z	Leave open	Has internal 50k pull up
TDI	I	3V3	Z	Leave open	Has internal 50k pull up
TDO	O	3V3	O	Leave open	Has internal 50k pull up

^a The PDN column indicates power-down state (when RUN pin LOW)

^b Must be driven by an open-collector driver

^c GPIO have software enabled pulls which keep state over power-down

^d Only available on Lite variants

^e The CM will always try to boot from this interface first

^f Requires external pull-up resistor to 5V as per HDMI spec

Table 3: Pin Functions



6 Electrical Specification

Caution! Stresses above those listed in Table 4 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VBAT	Core SMPS Supply	-0.5	6.0	V
3V3	3V3 Supply Voltage	-0.5	4.10	V
1V8	1V8 Supply Voltage	-0.5	2.10	V
VDAC	TV DAC Supply	-0.5	4.10	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	-0.5	4.10	V
GPIO28-45_VDD	GPIO28-45 I/O Supply Voltage	-0.5	4.10	V
SDX_VDD	Primary SD/eMMC Supply Voltage	-0.5	4.10	V

Table 4: Absolute Maximum Ratings

DC Characteristics are defined in Table 5



Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage ^a	VDD_IO = 1.8V	-	-	0.6	V
		VDD_IO = 2.7V	-	-	0.8	V
		VDD_IO = 3.3V	-	-	0.9	V
V_{IH}	Input high voltage ^a	VDD_IO = 1.8V	1.0	-	-	V
		VDD_IO = 2.7V	1.3	-	-	V
		VDD_IO = 3.3V	1.6	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	5	μA
C_{IN}	Input capacitance	-	-	5	-	pF
V_{OL}	Output low voltage ^b	VDD_IO = 1.8V, IOL = -2mA	-	-	0.2	V
		VDD_IO = 2.7V, IOL = -2mA	-	-	0.15	V
		VDD_IO = 3.3V, IOL = -2mA	-	-	0.14	V
V_{OH}	Output high voltage ^b	VDD_IO = 1.8V, IOH = 2mA	1.6	-	-	V
		VDD_IO = 2.7V, IOH = 2mA	2.5	-	-	V
		VDD_IO = 3.3V, IOH = 2mA	3.0	-	-	V
I_{OL}	Output low current ^c	VDD_IO = 1.8V, VO = 0.4V	12	-	-	mA
		VDD_IO = 2.7V, VO = 0.4V	17	-	-	mA
		VDD_IO = 3.3V, VO = 0.4V	18	-	-	mA
I_{OH}	Output high current ^c	VDD_IO = 1.8V, VO = 1.4V	10	-	-	mA
		VDD_IO = 2.7V, VO = 2.3V	16	-	-	mA
		VDD_IO = 3.3V, VO = 2.3V	17	-	-	mA
R_{PU}	Pullup resistor	-	50	-	65	kΩ
R_{PD}	Pulldown resistor	-	50	-	65	kΩ

^a Hysteresis enabled

^b Default drive strength (8mA)

^c Maximum drive strength (16mA)

Table 5: DC Characteristics

AC Characteristics are defined in Table 6 and Fig. 3.

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time ^a	-	1.6	-	ns
Digital outputs	t_{fall}	90-10% fall time ^a	-	1.7	-	ns
GPCLK	t_{JOSC}	Oscillator-derived GPCLK cycle-cycle jitter (RMS)	-	-	20	ps
GPCLK	t_{JPLL}	PLL-derived GPCLK cycle-cycle jitter (RMS)	-	-	48	ps

^a Default drive strength, CL = 5pF, VDD_IOx = 3.3V

Table 6: Digital I/O Pin AC Characteristics

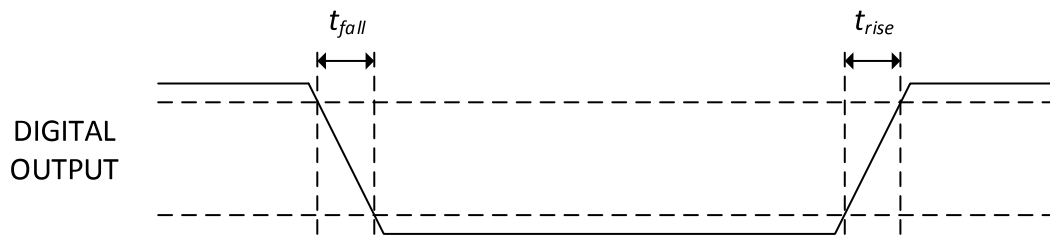


Figure 3: Digital IO Characteristics

7 Power Supplies

The Compute Module 3+ has six separate supplies that must be present and powered at all times; you cannot leave any of them unpowered, even if a specific interface or GPIO bank is unused. The six supplies are as follows:

1. VBAT is used to power the BCM2837 processor core. It feeds the SMPS that generates the chip core voltage.
2. 3V3 powers various BCM2837 PHYs, IO and the eMMC Flash.
3. 1V8 powers various BCM2837 PHYs, IO and SDRAM.
4. VDAC powers the composite (TV-out) DAC.
5. GPIO0-27_VREF powers the GPIO 0-27 IO bank.
6. GPIO28-45_VREF powers the GPIO 28-45 IO bank.

Supply	Description	Minimum	Typical	Maximum	Unit
VBAT	Core SMPS Supply	2.5	-	5.0 + 5%	V
3V3	3V3 Supply Voltage	3.3 - 5%	3.3	3.3 + 5%	V
1V8	1V8 Supply Voltage	1.8 - 5%	1.8	1.8 + 5%	V
VDAC	TV DAC Supply ^a	2.5 - 5%	2.8	3.3 + 5%	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
GPIO28-45_VDD	GPIO28-45 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
SDX_VDD	Primary SD/eMMC Supply Voltage	1.8 - 5%	-	3.3 + 5%	V

^a Requires a clean 2.5-2.8V supply if TV DAC is used, else connect to 3V3

Table 7: Power Supply Operating Ranges



7.1 Supply Sequencing

Supplies should be staggered so that the highest voltage comes up first, then the remaining voltages in descending order. This is to avoid forward biasing internal (on-chip) diodes between supplies, and causing latch-up. Alternatively supplies can be synchronised to come up at exactly the same time as long as at no point a lower voltage supply rail voltage exceeds a higher voltage supply rail voltage.

7.2 Power Requirements

Exact power requirements will be heavily dependent upon the individual use case. If an on-chip subsystem is unused, it is usually in a low power state or completely turned off. For instance, if your application does not use 3D graphics then a large part of the core digital logic will never turn on and need power. This is also the case for camera and display interfaces, HDMI, USB interfaces, video encoders and decoders, and so on.

Powerchain design is critical for stable and reliable operation of the Compute Module 3+. We strongly recommend that designers spend time measuring and verifying power requirements for their particular use case and application, as well as paying careful attention to power supply sequencing and maximum supply voltage tolerance.

Table 8 specifies the recommended minimum power supply outputs required to power the Compute Module 3+.

Supply	Minimum Requirement	Unit
VBAT (CM1)	2000 ^a	mW
VBAT (CM3,3L)	3500 ^a	mW
3V3	250	mA
1V8	250	mA
VDAC	25	mA
GPIO0-27_VDD	50 ^b	mA
GPIO28-45_VDD	50 ^b	mA
SDX_VDD	50 ^b	mA

^a Recommended minimum. Actual power drawn is very dependent on use-case

^b Each GPIO can supply up to 16mA, aggregate current per bank must not exceed 50mA

Table 8: Minimum Power Supply Requirements

8 Booting

The eMMC Flash device on CM3+ is directly connected to the primary BCM2837 SD/eMMC interface. These connections are not accessible on the module pins. On CM3+ Lite this SD interface is available on the SDX_ pins.



When initially powered on, or after the RUN pin has been held low and then released, the BCM2837 will try to access the primary SD/eMMC interface. It will then look for a file called bootcode.bin on the primary partition (which must be FAT) to start booting the system. If it cannot access the SD/eMMC device or the boot code cannot be found, it will fall back to waiting for boot code to be written to it over USB; in other words, its USB port is in slave mode waiting to accept boot code from a suitable host.

A USB boot tool is available on Github which allows a host PC running Linux to write the BCM2837 boot code over USB to the module. That boot code then runs and provides access to the SD/eMMC as a USB mass storage device, which can then be read and written using the host PC. Note that a Raspberry Pi can be used as the host machine. For those using Windows a precompiled and packaged tool is available. For more information see [here](#).

The Compute Module has a pin called EMMC_DISABLE_N which when shorted to GND will disable the SD/eMMC interface (by physically disconnecting the SD_CMD pin), forcing BCM2837 to boot from USB. Note that when the eMMC is disabled in this way, it takes a couple of seconds from powering up for the processor to stop attempting to talk to the SD/eMMC device and fall back to booting from USB.

Note that once booted over USB, BCM2837 needs to re-enable the SD/eMMC device (by releasing EMMC_DISABLE_N) to allow access to it as mass storage. It expects to be able to do this by driving the EMMC_EN_N_1V8 pin LOW, which at boot is initially an input with a pull up to 1V8. If an end user wishes to add the ability to access the SD/eMMC over USB in their product, similar circuitry to that used on the Compute Module IO Board to enable/disable the USB boot and SD/eMMC must be used; that is, EMMC_DISABLE_N pulled low via MOSFET(s) and released again by MOSFET, with the gate controlled by EMMC_EN_N_1V8. **Ensure you use MOSFETs suitable for switching at 1.8V (i.e. use a device with gate threshold voltage, V_t , suitable for 1.8V switching).**

9 Peripherals

9.1 GPIO

BCM2837 has in total 54 GPIO lines in 3 separate voltage banks. All GPIO pins have at least two alternative functions within the SoC. When not used for the alternate peripheral function, each GPIO pin may be set as an input (optionally as an interrupt) or an output. The alternate functions are usually peripheral I/Os, and most peripherals appear twice to allow flexibility on the choice of I/O voltage.

GPIO bank2 is used on the module to connect to the eMMC device and for an on-board I2C bus (to talk to the core SMPS and control the special function pins). On CM3+ Lite most of bank2 is exposed to allow a user to connect their choice of SD card or eMMC device (if required).

Bank0 and 1 GPIOs are available for general use. GPIO0 to GPIO27 are bank0 and GPIO28-45 make up bank1. GPIO0-27_VDD is the power supply for bank0 and GPIO28-45_VDD is the power supply for bank1. SDX_VDD is the supply for bank2 on CM3+ Lite. These supplies can be in the range 1.8V-3.3V (see Table 7) and are not optional; each bank must be powered, even when none of the GPIOs for that bank are used.

Note that the HDMI_HPD_N_1V8 and EMMC_EN_N_1V8 pins are 1.8V IO and are used for special functions (HDMI hot plug detect and boot control respectively). Please do not use these pins for any other purpose, as the software for the module will always expect these pins to have these special functions. If they are unused please leave them unconnected.



All GPIOs except GPIO28, 29, 44 and 45 have weak in-pad pull-ups or pull-downs enabled when the device is powered on. It is recommended to add off-chip pulls to GPIO28, 29, 44 and 45 to make sure they never float during power on and initial boot.

9.1.1 GPIO Alternate Functions

GPIO	Default						
	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	-	-	-
1	High	SCL0	SA4	DE	-	-	-
2	High	SDA1	SA3	LCD_VSYNC	-	-	-
3	High	SCL1	SA2	LCD_HSYNC	-	-	-
4	High	GPCLK0	SA1	DPL.D0	-	-	ARM_TDI
5	High	GPCLK1	SA0	DPL.D1	-	-	ARM_TDO
6	High	GPCLK2	SOE_N	DPL.D2	-	-	ARM_RTCK
7	High	SPI0_CE1_N	SWE_N	DPL.D3	-	-	-
8	High	SPI0_CE0_N	SD0	DPL.D4	-	-	-
9	Low	SPI0_MISO	SD1	DPL.D5	-	-	-
10	Low	SPI0_MOSI	SD2	DPL.D6	-	-	-
11	Low	SPI0_SCLK	SD3	DPL.D7	-	-	-
12	Low	PWM0	SD4	DPL.D8	-	-	ARM_TMS
13	Low	PWM1	SD5	DPL.D9	-	-	ARM_TCK
14	Low	TXD0	SD6	DPL.D10	-	-	TXD1
15	Low	RXD0	SD7	DPL.D11	-	-	RXD1
16	Low	FL0	SD8	DPL.D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPL.D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPL.D14	-	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPL.D15	-	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPL.D16	-	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPL.D17	-	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPL.D18	SD1_CLK	ARM_TRST	-
23	Low	SD0_CMD	SD15	DPL.D19	SD1_CMD	ARM_RTCK	-
24	Low	SD0_DAT0	SD16	DPL.D20	SD1_DAT0	ARM_TDO	-
25	Low	SD0_DAT1	SD17	DPL.D21	SD1_DAT1	ARM_TCK	-
26	Low	SD0_DAT2	TE0	DPL.D22	SD1_DAT2	ARM_TDI	-
27	Low	SD0_DAT3	TE1	DPL.D23	SD1_DAT3	ARM_TMS	-

Table 9: GPIO Bank0 Alternate Functions



GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
28	None	SDA0	SA5	PCM_CLK	FL0	-	-
29	None	SCL0	SA4	PCM_FS	FL1	-	-
30	Low	TE0	SA3	PCM_DIN	CTS0	-	CTS1
31	Low	FL0	SA2	PCM_DOUT	RTS0	-	RTS1
32	Low	GPCLK0	SA1	RING_OCLK	TXD0	-	TXD1
33	Low	FL1	SA0	TE1	RXD0	-	RXD1
34	High	GPCLK0	SOE_N	TE2	SD1_CLK	-	-
35	High	SPI0_CE1_N	SWE_N	-	SD1_CMD	-	-
36	High	SPI0_CE0_N	SD0	TXD0	SD1_DAT0	-	-
37	Low	SPI0_MISO	SD1	RXD0	SD1_DAT1	-	-
38	Low	SPI0_MOSI	SD2	RTS0	SD1_DAT2	-	-
39	Low	SPI0_SCLK	SD3	CTS0	SD1_DAT3	-	-
40	Low	PWM0	SD4	-	SD1_DAT4	SPI2_MISO	TXD1
41	Low	PWM1	SD5	TE0	SD1_DAT5	SPI2_MOSI	RXD1
42	Low	GPCLK1	SD6	TE1	SD1_DAT6	SPI2_SCLK	RTS1
43	Low	GPCLK2	SD7	TE2	SD1_DAT7	SPI2_CE0_N	CTS1
44	None	GPCLK1	SDA0	SDA1	TE0	SPI2_CE1_N	-
45	None	PWM1	SCL0	SCL1	TE1	SPI2_CE2_N	-

Table 10: GPIO Bank1 Alternate Functions

Table 9 and Table 10 detail the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the Broadcom Peripherals Specification document and have Linux drivers available.

9.1.2 Secondary Memory Interface (SMI)

The SMI peripheral is an asynchronous NAND type bus supporting Intel mode80 type transfers at 8 or 16 bit widths and available in the ALT1 positions on GPIO banks 0 and 1 (see Table 9 and Table 10). It is not publicly documented in the Broadcom Peripherals Specification but a Linux driver is available in the Raspberry Pi Github Linux repository (`bcm2835_smi.c` in `linux/drivers/misc`).

9.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available on bank 0 GPIOs. This up-to-24-bit parallel interface can support a secondary display. Again this interface is not documented in the Broadcom Peripherals Specification but documentation can be found [here](#).



9.1.4 SD/SDIO Interface

The BCM283x supports two SD card interfaces, SD0 and SD1.

The first (SD0) is a proprietary Broadcom controller that does not support SDIO and is the primary interface used to boot and talk to the eMMC or SDX_x signals.

The second interface (SD1) is standards compliant and can interface to SD, SDIO and eMMC devices; for example on a Raspberry Pi 3 B+ it is used to talk to the on-board CYW43455 WiFi device in SDIO mode.

Both interfaces can support speeds up to 50MHz single ended (SD High Speed Mode).

9.2 CSI (MIPI Serial Camera)

Currently the CSI interface is not openly documented and only CSI camera sensors supported by the official Raspberry Pi firmware will work with this interface. Supported sensors are the OmniVision OV5647 and Sony IMX219.

It is recommended to attach other cameras via USB.

9.3 DSI (MIPI Serial Display)

Currently the DSI interface is not openly documented and only DSI displays supported by the official Raspberry Pi firmware will work with this interface.

Displays can also be added via the parallel DPI interface which is available as a GPIO alternate function - see Table 9 and Section 9.1.3

9.4 USB

The BCM2837 USB port is On-The-Go (OTG) capable. If using either as a fixed slave or fixed master, please tie the USB_OTGID pin to ground.

The USB port (Pins USB_DP and USB_DM) must be routed as 90 ohm differential PCB traces.

Note that the port is capable of being used as a true OTG port however there is no official documentation. Some users have had success making this work.

9.5 HDMI

BCM283x supports HDMI V1.3a.

It is recommended that users follow a similar arrangement to the Compute Module IO Board circuitry for HDMI output.

The HDMI CK_P/N (clock) and D0-D2_P/N (data) pins must each be routed as matched length 100 ohm differential PCB traces. It is also important to make sure that each differential pair is closely phase matched. Finally, keep HDMI traces well away from other noise sources and as short as possible.

Failure to observe these design rules is likely to result in EMC failure.



9.6 Composite (TV Out)

The TVDAC pin can be used to output composite video (PAL or NTSC). Please route this signal away from noise sources and use a 75 ohm PCB trace.

Note that the TV DAC is powered from the VDAC supply which must be a clean supply of 2.5-2.8V. It is recommended users generate this supply from 3V3 using a low noise LDO.

If the TVDAC output is not used VDAC can be connected to 3V3, but it must be powered even if the TV-out functionality is unused.

10 Thermals

The BCM2837 SoC employs DVFS (Dynamic Voltage and Frequency Scaling) on the core voltage. When the processor is idle (low CPU utilisation), it will reduce the core frequency and voltage to reduce current draw and heat output. When the core utilisation exceeds a certain threshold the core voltage is increased and the core frequency is boosted to the maximum working frequency of 1.2GHz. The voltage and frequency are throttled back when the CPU load reduces back to an 'idle' level OR when the silicon temperature as measured by the on-chip temperature sensor exceeds 80C (thermal throttling).

A designer must pay careful attention to the thermal design of products using the CM3+ so that performance is not artificially curtailed due to the processor thermal throttling, as the Quad ARM complex in the BCM2837 can generate significant heat output under load.

10.1 Temperature Range

The operating temperature range of the module is set by the lowest maximum and highest minimum of any of the components used.

The eMMC and LPDDR2 have the narrowest range, these are rated for -25 to +80 degrees Celsius. Therefore the nominal range for the CM3+ and CM3+ Lite is -25C to +80C.

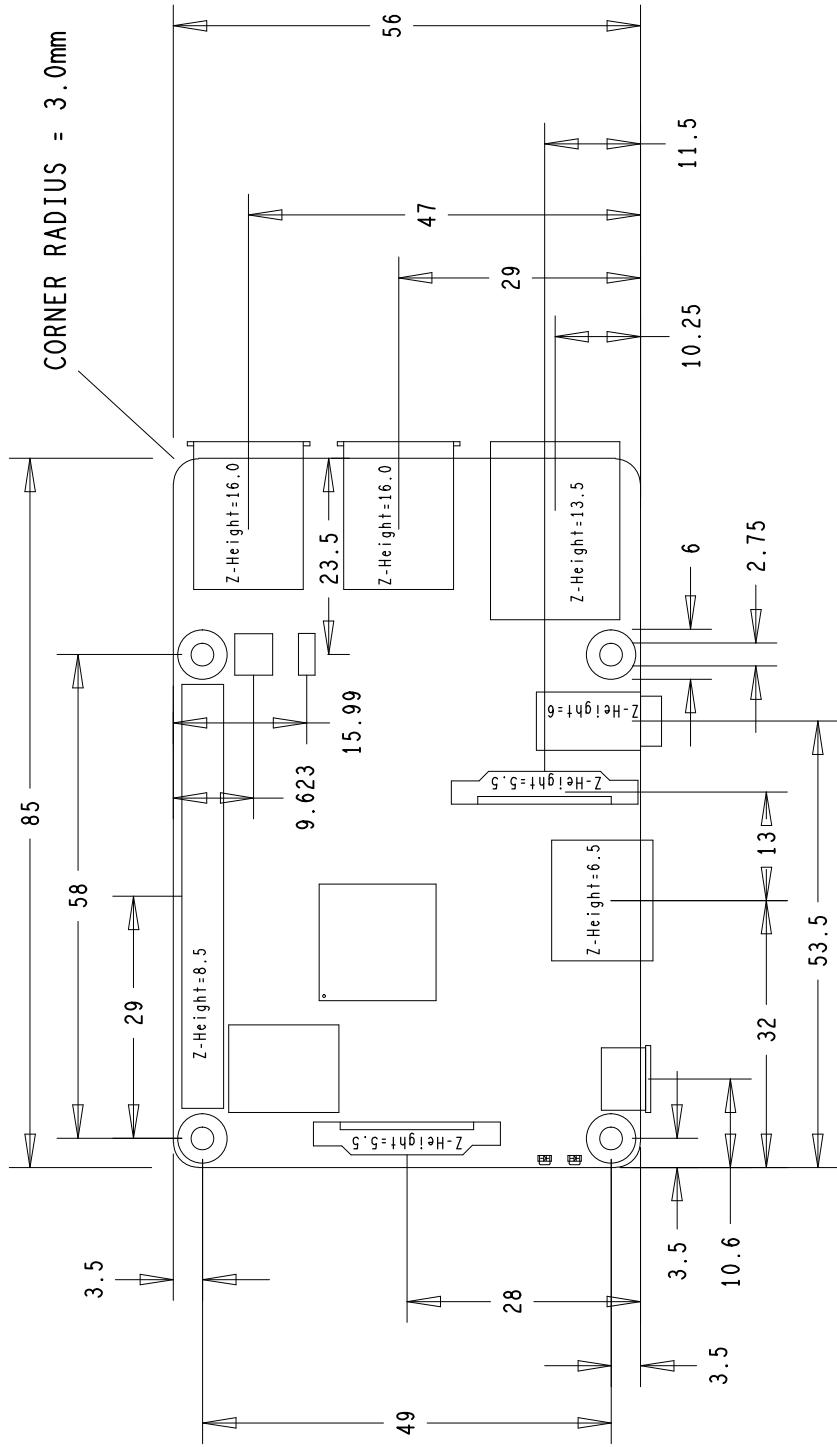
However, this range is the maximum for the silicon die; therefore, users would have to take into account the heat generated when in use and make sure this does not cause the temperature to exceed 80 degrees Celsius.

11 Availability

Raspberry Pi guarantee availability of CM3+ and CM3+ Lite until at least January 2026.

12 Support

For support please see the hardware documentation section of the Raspberry Pi website and post questions to the Raspberry Pi forum.



Annex 2

A2.1 Software design principles and strategies

The work on specific software implementation aimed to three core research products:

- A JSON schema based on AGS exchange format for sensor-enabled geosynthetics
- A FEA-driven SAAS for smart foundations
- A set of tools for remote sensing for smart geocells

The (custom) JSON format is built on top of the general and well-known JSON standard as outlined <https://www.json.org/json-it.html>. JSON is based on two main structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

The dictionary structure, in this specific case, is straightforwardly derived from the AGS NZ 1.0.1 format (2017).

All software designed, drafted and produced during the R&D process has been written primarily in Python 3, with the exception of Docker CLI and Code Aster specific command lines. The choice of this particular language is dictated by the current internal IT management system (including FEA, GIS and BIM strategic protocols) and the writer's own technical background.

The main challenge of the SAAS product design (smart foundations) was to integrate a rather complex, non linear 3D FEA procedure into a backend REST API that should operate quickly (<20 s for each POST call) and scale on approx. 5000 calls per day. This was achieved by:

- Using a lightweight, custom-made Linux Docker base image and an industry-level Docker-Kubernetes capable PAAS.
- Optimizing the automatic pre-processing (geometry, grouping and meshing) by bootstrapping and simplifying the standard procedures implemented in the Salome Platform (see table below for references).
- Optimizing the FEA and post-processing procedures built on top of Code Aster (see table below for references)

On the other hand, the main goal of the sensing-related software (generally smart geocells) was to reduce the edge computing costs at minimum while keeping the ability to manipulate and transform the data to the desired extent. For this sake, the CircuitPython, Pandas, Scipy and Scikit-Learn libraries proved more than adequate (see Table 1 for references).

Finally, as for the data security layer prototype, a light implementation of a permissioned (private) blockchain based on the Hyperledger Iroha framework (Python SDK) was performed.

Several third-party platforms and libraries were used to produce the software. Table 1 reports a list of the most important software.

Platform/Library	Description	Source
Salome	The SALOME platform is an open source software framework for the integration of numerical solvers in various scientific domains. SALOME is integrating a CAD/CAE modeling tool, industrial mesh generators, and advanced 3D visualization features.	https://www.salome-platform.org/ (GNU LGPL)
Code Aster	Code Aster offers a full range of multiphysical analysis and modelling methods in the fields of thermo-mechanics. Its modelling, algorithms and solvers are constantly under construction to improve and complete them (1,200,000 lines of code, 200 operators). It is a fully open-source platform and it is linked, coupled and encapsulated in numerous ways.	https://www.code-aster.org/ (GNU LGPL)
Anaconda	Anaconda is an open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.). The distribution includes data-science packages suitable for Windows, Linux, and macOS. It includes popular libraries such as Numpy, Scipy, Pandas, Scikit-learn, Theano, PyTorch, etc.	https://www.anaconda.com/ (All rights reserved under the 3-clause BSD License)
Docker	Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called	https://www.docker.com/ (Apache License 2.0)

	<p>containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating system kernel and therefore use fewer resources than virtual machines.</p>	
Kubernetes	<p>Kubernetes is an open-source container-orchestration system for automating computer application deployment, scaling, and management.</p> <p>It was originally designed by Google and is now maintained by the Cloud Native Computing Foundation. It aims to provide a "platform for automating deployment, scaling, and operations of application containers across clusters of hosts". It works with a range of container tools, including Docker</p>	<p>https://kubernetes.io/ (Apache License 2.0)</p>
Flask	<p>Flask is a micro web framework written in Python. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.</p>	<p>https://flask.palletsprojects.com/ (All rights reserved under the 3-clause BSD License)</p>
CircuitPython	<p>CircuitPython is an open source derivative of the MicroPython programming language. Development of CircuitPython is supported by Adafruit Industries. It is a software implementation of the Python 3 programming language, written in C. It has been ported to run on several modern microcontrollers. CircuitPython is a full Python compiler and runtime that runs on the microcontroller hardware. Included are a selection of core Python libraries. CircuitPython includes modules which give the programmer access to</p>	<p>https://circuitpython.org/ (MIT license)</p>

	the low-level hardware of Adafruit compatible products as well as a number of higher level libraries.	
Hyperledger Iroha	Hyperledger Iroha is a general purpose permissioned blockchain system that can be used to manage digital assets, identity, and serialized data. This can be useful for applications such as interbank settlement, central bank digital currencies, payment systems, national IDs, and logistics, among others.	https://www.hyperledger.org/use/iroha (Apache License 2.0)

Table 1

A2.2 JSON schema

The proposed JSON schema is intended to be compatible with any possible inertial sensor configuration for a generic sensor-enabled geosynthetic. The same structure can be easily extended to other sensor-enabled geotechnical object and/or sensor types. As anticipated, the schema is built on top of AGS NZ ver. 1.0.1 (2017) and should be fully compatible with the latest release of AGS ver. 4.1 (December 2020). The schema implementation for the proposed format is the draft-07.

```

{
  "$id": "",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "PROJ": {
    "PROJ_ID": {
      "type": "string"
    },
    "PROJ_NAME": {
      "type": "string"
    },
    "PROJ_CLNT": {
      "type": "string"
    },
    "PROJ_CONT": {
      "type": "string"
    },
    "PROJ_ENG": {
      "type": "string"
    },
    "LOCA": {
      "LOCA_ID": {
        "type": "string"
      }
    }
  }
}

```



```

"LOCA_TYPE": {
  "type": "string"
},
"LOCA_STAT": {
  "type": "string"
},
"LOCA_LAT": {
  "type": "number"
},
"LOCA_LON": {
  "type": "number"
},
"LOCA_GL": {
  "type": "number"
},
"LOCA_LLZ": {
  "type": "string"
},
"LOCA_REM": {
  "type": "string"
},
"MONG": {
  "MONG_ID": {
    "type": "string"
  },
  "MONG_DIS": {
    "type": "number"
  },
  "MONG_DATE": {
    "type": "date string"
  },
  "MONG_TYPE": {
    "type": "string"
  },
  "MONG_CONT": {
    "type": "string"
  },
  "MOND": {
    "MOND_DTIM": {
      "type": "array",
      "items": {
        "type": "datetime"
      }
    },
  },
  "MOND_RDNG": {
    "type": "array",

```

```

        "items": {
            "type": "string"
        },
        "MOND_TYPE": {
            "type": "string"
        }
    }
}
}
}

```

A2.3 Working with containers

The Docker community official websites provides the following definition of a container: [...] *A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.*

For the proposed implementations, Docker was found to be a convenient framework on top of which most of the cloud-based prototypes and products were developed. Docker comes with its own CLI and rules to build and maintain containers.

The following example shows one “dockerfile” used to set up the calculation environment for one of the cloud-based FEA prototypes.

```

# base Linux image pull(omissis)

# Setup ENV
ENV TZ=Europe/Rome
ENV PORT 8080
ENV HOST 0.0.0.0

# Echo Timezone
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

# Install extra debian packages and python modules
RUN export DEBIAN_FRONTEND=noninteractive && \
    apt-get update && \
    apt install software-properties-common -y && \
    apt-get install python3-pip -y && \
    python3 -m pip install numpy==1.18.0 && \
    python3 -m pip install pandas && \

```

```

python3 -m pip install flask && \
python3 -m pip install matplotlib && \
python3 -m pip install scipy && \
rm -rf /var/lib/apt/lists/*

WORKDIR /home/aster/shared

# Copy files from local repo
COPY aster_temp app.py ./

ENTRYPOINT ["python3"]

CMD ["app.py"]

```

A standard dockerfile is made up of several simple CLI commands that are used to e.g. set environmental variables (ENV), run specific commands (RUN), copy files from different file sources (COPY) and execute loaded applications (CMD).

In the specific case, the Docker CLI was used to instruct the cloud-based service (which provided a Kubernetes-based docker orchestration protocol) to pull a tailor-made lightweight Debian (Linux) image, to initialize ports and other environmental variables compatibly with the network architecture, to install all the necessary python3 modules and finally to load and execute one of our backend applications.

A2.4 Automation in Salome

Automation in Salome is carried out through its built-in Python Interface. All Salome commands are accessible through such interface and can be used to perform deep automation on geometry creation and manipulation as well as meshing and other FEA-related tasks.

In this project several tools to automate geometry and meshing creation were developed for assisting different FEA cloud-based procedures. The following example shows part of the code used to automate the creation of a regular (grid-like) mesh representing a stiffened raft on swelling soil.

```

### BUILD GEOMETRY ###

for i in s_x_r:

    for j in s_y_r:

        if i>=j:

            for D in D_cr:

```

```

m_x = (1.5 * i*rib_L) / D
m_y = (1.5 * j*rib_L) / D

geompy = geomBuilder.New(theStudy)

O = geompy.MakeVertex(0, 0, 0)
OX = geompy.MakeVectorDXDYDZ(1, 0, 0)
OY = geompy.MakeVectorDXDYDZ(0, 1, 0)
OZ = geompy.MakeVectorDXDYDZ(0, 0, 1)
Vertex_1 = geompy.MakeVertex(0, 0, 0)
Multi_Translation_1 = geompy.MakeMultiTranslation1D(Vertex_1,
OX, rib_L, i+1)

Multi_Translation_2 = geompy.MakeMultiTranslation1D(Vertex_1,
OY, rib_L, j+1)

Multi_Translation_3 =
geompy.MakeMultiTranslation1D(Multi_Translation_1, OY, 0.5*rib_L, 2*j+1)
Multi_Translation_4 =
geompy.MakeMultiTranslation1D(Multi_Translation_2, OX, 0.5*rib_L, 2*i+1)
Grid = geompy.MakeFuseList([Multi_Translation_3,
Multi_Translation_4], True, True)

Base_grid = geompy.MakeTranslation(Grid, 0, 0, -1)
Vertex_2 = geompy.MakeTranslation(Vertex_1, 0, 0, -1)
Line_1 = geompy.MakeLineTwoPnt(Vertex_1, Vertex_2)
Multi_Translation_5 = geompy.MakeMultiTranslation1D(Line_1,
OX, rib_L, i+1)

Multi_Translation_6 = geompy.MakeMultiTranslation1D(Line_1,
OY, rib_L, j+1)

Multi_Translation_7 =
geompy.MakeMultiTranslation1D(Multi_Translation_5, OY, 0.5*rib_L, 2*j+1)
Multi_Translation_8 =
geompy.MakeMultiTranslation1D(Multi_Translation_6, OX, 0.5*rib_L, 2*i+1)
Springs = geompy.MakeFuseList([Multi_Translation_7,
Multi_Translation_8], True, True)

Vertex_3 = geompy.MakeTranslation(Vertex_1, rib_L, 0, 0)
Line_2 = geompy.MakeLineTwoPnt(Vertex_1, Vertex_3)
Vertex_4 = geompy.MakeTranslation(Vertex_1, 0, rib_L, 0)
Line_3 = geompy.MakeLineTwoPnt(Vertex_1, Vertex_4)
Multi_Translation_9 = geompy.MakeMultiTranslation2D(Line_2,
OX, rib_L, i, OY, rib_L, j+1)
Multi_Translation_10 = geompy.MakeMultiTranslation2D(Line_3,
OX, rib_L, i+1, OY, rib_L, j)

Partition_1 = geompy.MakePartition([Multi_Translation_9],
[Grid])

Partition_2 = geompy.MakePartition([Multi_Translation_10],
[Grid])

Partition_all = geompy.MakeFuse(Partition_1, Partition_2)

```

```

Corner_springs = geompy.CreateGroup (Springs,
geompy.ShapeType ["EDGE"])
Edge_springs = geompy.CreateGroup (Springs,
geompy.ShapeType ["EDGE"])
Center_springs_1 = geompy.CreateGroup (Springs,
geompy.ShapeType ["EDGE"])
Center_springs_2 = geompy.CreateGroup (Springs,
geompy.ShapeType ["EDGE"])
Center_springs_3 = geompy.CreateGroup (Springs,
geompy.ShapeType ["EDGE"])
Center_springs_4 = geompy.CreateGroup (Springs,
geompy.ShapeType ["EDGE"])
Base_nodes = geompy.CreateGroup (Springs,
geompy.ShapeType ["VERTEX"])

Cent_beams_x = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["EDGE"])
Cent_beams_y = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["EDGE"])
Edge_b_x_1 = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["EDGE"])
Edge_b_x_2 = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["EDGE"])
Edge_b_y_1 = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["EDGE"])
Edge_b_y_2 = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["EDGE"])
Res_c_x = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["VERTEX"])
Res_c_y = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["VERTEX"])
Res_e_x_1 = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["VERTEX"])
Res_e_x_2 = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["VERTEX"])
Res_e_y_1 = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["VERTEX"])
Res_e_y_2 = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["VERTEX"])
Res_all = geompy.CreateGroup (Partition_all,
geompy.ShapeType ["VERTEX"])

Springs_coords=[]
S_node_coords=[]

```

```

        for k_edge_ID in geompy.SubShapeAllSortedCentresIDs(Springs,
geompy.ShapeType["EDGE"]):
            k_edge=geompy.GetSubShape(Springs, [k_edge_ID])
            CDG=geompy.MakeCDG(k_edge)
            Springs_coords=geompy.PointCoordinates(CDG)
            if (Springs_coords[0]==i*rib_L and
Springs_coords[1]==j*rib_L) or (Springs_coords[0]==0 and Springs_coords[1]==0) or
(Springs_coords[0]==i*rib_L and Springs_coords[1]==0) or (Springs_coords[0]==0 and
Springs_coords[1]==j*rib_L):
                geompy.AddObject(Corner_springs, k_edge_ID)
            elif (Springs_coords[0]==i*rib_L and
Springs_coords[1]!=j*rib_L) or (Springs_coords[0]!=i*rib_L and
Springs_coords[1]==j*rib_L) or (Springs_coords[0]==0 and Springs_coords[1]!=0) or
(Springs_coords[0]!=0 and Springs_coords[1]==0):
                geompy.AddObject(Edge_springs, k_edge_ID)
            else:
                if (((2*abs(Springs_coords[1]-
0.5*j*rib_L)/(j*rib_L)**m_y*(1-(2*abs(Springs_coords[0]-
0.5*i*rib_L)/(i*rib_L)**m_x)+(2*abs(Springs_coords[0]-
0.5*i*rib_L)/(i*rib_L)**m_x))<0.05):
                    geompy.AddObject(Center_springs_1,
k_edge_ID)
                elif (((2*abs(Springs_coords[1]-
0.5*j*rib_L)/(j*rib_L)**m_y*(1-(2*abs(Springs_coords[0]-
0.5*i*rib_L)/(i*rib_L)**m_x)+(2*abs(Springs_coords[0]-
0.5*i*rib_L)/(i*rib_L)**m_x))<0.20):
                    geompy.AddObject(Center_springs_2,
k_edge_ID)
                elif (((2*abs(Springs_coords[1]-
0.5*j*rib_L)/(j*rib_L)**m_y*(1-(2*abs(Springs_coords[0]-
0.5*i*rib_L)/(i*rib_L)**m_x)+(2*abs(Springs_coords[0]-
0.5*i*rib_L)/(i*rib_L)**m_x))<0.50):
                    geompy.AddObject(Center_springs_3,
k_edge_ID)
                else:
                    geompy.AddObject(Center_springs_4,
k_edge_ID)

        for k_node_ID in geompy.SubShapeAllIDs(Springs,
geompy.ShapeType["VERTEX"]):
            k_node=geompy.GetSubShape(Springs, [k_node_ID])
            S_node_coords=geompy.PointCoordinates(k_node)
            if S_node_coords[2]!=0:
                geompy.AddObject(Base_nodes, k_node_ID)

```

```

Part_all_coords=[]
R_node_coords=[]

for t_edge_ID in
geompy.SubShapeAllSortedCentresIDs(Partition_all, geompy.ShapeType["EDGE"]):
    t_edge=geompy.GetSubShape(Partition_all, [t_edge_ID])
    CDG=geompy.MakeCDG(t_edge)
    Part_all_coords=geompy.PointCoordinates(CDG)
    if (Part_all_coords[0]==0):
        geompy.AddObject(Edge_b_y_1, t_edge_ID)
    elif (Part_all_coords[1]==0):
        geompy.AddObject(Edge_b_x_1, t_edge_ID)
    elif (Part_all_coords[0]==i*rib_L):
        geompy.AddObject(Edge_b_y_2, t_edge_ID)
    elif (Part_all_coords[1]==j*rib_L):
        geompy.AddObject(Edge_b_x_2, t_edge_ID)
    else:
        for jj in range(1,j):
            if (Part_all_coords[1]==jj*rib_L):
                geompy.AddObject(Cent_beams_x,
t_edge_ID)

            for ii in range(1,i):
                if (Part_all_coords[0]==ii*rib_L):
                    geompy.AddObject(Cent_beams_y,
t_edge_ID)

for t_node_ID in geompy.SubShapeAllIDs(Partition_all,
geompy.ShapeType["VERTEX"]):
    t_node=geompy.GetSubShape(Partition_all, [t_node_ID])
    R_node_coords=geompy.PointCoordinates(t_node)
    geompy.AddObject(Res_all, t_node_ID)
    if (R_node_coords[1]==0):
        geompy.AddObject(Res_e_x_1, t_node_ID)
    elif (R_node_coords[0]==0):
        geompy.AddObject(Res_e_y_1, t_node_ID)
    elif (R_node_coords[1]==j*rib_L):
        geompy.AddObject(Res_e_x_2, t_node_ID)
    elif (R_node_coords[0]==i*rib_L):
        geompy.AddObject(Res_e_y_2, t_node_ID)

    for jj in range(1,j):
        if (R_node_coords[1]==jj*rib_L):
            geompy.AddObject(Res_c_x,
t_node_ID)

    for ii in range(1,i):
        if (R_node_coords[0]==ii*rib_L):

```

```

t_node_ID)
                                                    geompy.AddObject (Res_c_y,

id_group1 = geompy.addToStudy (Corner_springs,
"Corner_springs")
id_group2 = geompy.addToStudy (Edge_springs, "Edge_springs")
id_group3 = geompy.addToStudy (Center_springs_1,
"Center_springs_1")
id_group4 = geompy.addToStudy (Center_springs_2,
"Center_springs_2")
id_group5 = geompy.addToStudy (Center_springs_3,
"Center_springs_3")
id_group6 = geompy.addToStudy (Center_springs_4,
"Center_springs_4")

id_group7 = geompy.addToStudy (Cent_beams_x, "Cent_beams_x")
id_group8 = geompy.addToStudy (Cent_beams_y, "Cent_beams_y")
id_group9 = geompy.addToStudy (Edge_b_x_1, "Edge_b_x_1")
id_group10 = geompy.addToStudy (Edge_b_x_2, "Edge_b_x_2")
id_group11 = geompy.addToStudy (Edge_b_y_1, "Edge_b_y_1")
id_group12 = geompy.addToStudy (Edge_b_y_2, "Edge_b_y_2")
id_group13 = geompy.addToStudy (Base_nodes, "Base_nodes")
id_group14 = geompy.addToStudy (Res_all, "Res_all")
id_group15 = geompy.addToStudy (Res_e_x_1, "Res_e_x_1")
id_group16 = geompy.addToStudy (Res_e_x_2, "Res_e_x_2")
id_group17 = geompy.addToStudy (Res_e_y_1, "Res_e_y_1")
id_group18 = geompy.addToStudy (Res_e_y_2, "Res_e_y_2")
id_group19 = geompy.addToStudy (Res_c_x, "Res_c_x")
id_group20 = geompy.addToStudy (Res_c_y, "Res_c_y")

### Create mesh and mesh groups

smesh = smeshBuilder.New (theStudy)
Mesh_1 = smesh.Mesh (Partition_all)
Mesh_2 = smesh.Mesh (Springs)
Regular_1D = Mesh_1.Segment ()
Number_of_Segments_1 = Regular_1D.NumberOfSegments (1)
Regular_1D = Mesh_2.Segment ()
Number_of_Segments_2 = Regular_1D.NumberOfSegments (1)
Mesh_1.Compute ()
Mesh_2.Compute ()
smesh.SetName (Mesh_1, 'Mesh_1')
smesh.SetName (Mesh_2, 'Mesh_2')

corn_s =
Mesh_2.GroupOnGeom (Corner_springs, 'corn_s', SMESH.EDGE)
edge_s = Mesh_2.GroupOnGeom (Edge_springs, 'edge_s', SMESH.EDGE)

```



```

        cent_s_1
Mesh_2.GroupOnGeom(Center_springs_1, 'cent_s_1', SMESH.EDGE)
        cent_s_2
Mesh_2.GroupOnGeom(Center_springs_2, 'cent_s_2', SMESH.EDGE)
        cent_s_3
Mesh_2.GroupOnGeom(Center_springs_3, 'cent_s_3', SMESH.EDGE)
        cent_s_4
Mesh_2.GroupOnGeom(Center_springs_4, 'cent_s_4', SMESH.EDGE)
        cent_b_x
Mesh_1.GroupOnGeom(Cent_beams_x, 'cent_b_x', SMESH.EDGE)
        cent_b_y
Mesh_1.GroupOnGeom(Cent_beams_y, 'cent_b_y', SMESH.EDGE)
        edge_b_x1
Mesh_1.GroupOnGeom(Edge_b_x_1, 'edge_b_x1', SMESH.EDGE)
        edge_b_x2
Mesh_1.GroupOnGeom(Edge_b_x_2, 'edge_b_x2', SMESH.EDGE)
        edge_b_y1
Mesh_1.GroupOnGeom(Edge_b_y_1, 'edge_b_y1', SMESH.EDGE)
        edge_b_y2
Mesh_1.GroupOnGeom(Edge_b_y_2, 'edge_b_y2', SMESH.EDGE)

        base_n = Mesh_2.GroupOnGeom(Base_nodes, 'base_n', SMESH.NODE)
        res_all = Mesh_1.GroupOnGeom(Res_all, 'res_all', SMESH.NODE)
        res_ex1 = Mesh_1.GroupOnGeom(Res_e_x_1, 'res_ex1', SMESH.NODE)
        res_ex2 = Mesh_1.GroupOnGeom(Res_e_x_2, 'res_ex2', SMESH.NODE)
        res_ey1 = Mesh_1.GroupOnGeom(Res_e_y_1, 'res_ey1', SMESH.NODE)
        res_ey2 = Mesh_1.GroupOnGeom(Res_e_y_2, 'res_ey2', SMESH.NODE)
        res_cx = Mesh_1.GroupOnGeom(Res_c_x, 'res_cx', SMESH.NODE)
        res_cy = Mesh_1.GroupOnGeom(Res_c_y, 'res_cy', SMESH.NODE)
        C_Mesh = smesh.Concatenate([Mesh_1, Mesh_2], 1, 1, 1e-05,
True, name='C_Mesh')

### Save and export mesh in MED format
# (omissis)

```

The above code includes both commands from Salome's geometry module (GEOMPY) and commands from Salome's mesh module (SMESH).

A2.5 Automation in Code Aster

Automation in Code Aster is performed embedding Python code into the command (.comm) file. Such (ASCII) file is used to instruct Code Aster on the various phases of the FEA processing, including mesh manipulation, material properties and constitutive laws, boundary conditions, solvers and calculation parameters, output preparation and delivery. The command file is a hybrid instruction

file that includes code lines from the specific programming language and may include Python 3 code lines as well.

```
# Read mesh file (.med from Salome)
mesh = LIRE_MALLAGE(UNITE=2)

# Import relevant Python3 modules
import numpy as np

# (omissis)

# define extra groups on the mesh
for i in seq_x1:
    mesh = DEFI_GROUP(reuse=mesh,
                      CREA_GROUP_MA=_F(NOM='pile_'+str(i)+'_'+str(0),
                                         OPTION='SPHERE',
                                         POINT=(i, 0, -1.0),
                                         RAYON=0.2,
                                         TYPE_MAILLE='SEG2'),
                      MAILLAGE=mesh)

    pile_groups_x1.append('pile_'+str(i)+'_'+str(0))

mesh = DEFI_GROUP(reuse=mesh,
                  CREA_GROUP_MA=_F(NOM='pile_x1',
                                   UNION=pile_groups_x1),
                  MAILLAGE=mesh)

for i in seq_x2:
    mesh = DEFI_GROUP(reuse=mesh,
                      CREA_GROUP_MA=_F(NOM='pile_'+str(i)+'_'+str(L_y),
                                         OPTION='SPHERE',
                                         POINT=(i, L_y, -1.0),
                                         RAYON=0.2,
                                         TYPE_MAILLE='SEG2'),
                      MAILLAGE=mesh)

    pile_groups_x2.append('pile_'+str(i)+'_'+str(L_y))

mesh = DEFI_GROUP(reuse=mesh,
                  CREA_GROUP_MA=_F(NOM='pile_x2',
                                   UNION=pile_groups_x2),
                  MAILLAGE=mesh)

for j in seq_y1:
```

```

mesh = DEFI_GROUP (reuse=mesh,
                   CREA_GROUP_MA=_F (NOM='pile_'+str(0)+'_'+str(j),
                                     OPTION='SPHERE',
                                     POINT=(0, j, -1.0),
                                     RAYON=0.2,
                                     TYPE_MAILLE='SEG2'),
                   MAILLAGE=mesh)

pile_groups_y1.append('pile_'+str(0)+'_'+str(j))

mesh = DEFI_GROUP (reuse=mesh,
                   CREA_GROUP_MA=_F (NOM='pile_y1',
                                     UNION=pile_groups_y1),
                   MAILLAGE=mesh)

for j in seq_y2:
    mesh = DEFI_GROUP (reuse=mesh,
                       CREA_GROUP_MA=_F (NOM='pile_'+str(L_x)+'_'+str(j),
                                         OPTION='SPHERE',
                                         POINT=(L_x, j, -1.0),
                                         RAYON=0.2,
                                         TYPE_MAILLE='SEG2'),
                       MAILLAGE=mesh)

    pile_groups_y2.append('pile_'+str(L_x)+'_'+str(j))

mesh = DEFI_GROUP (reuse=mesh,
                   CREA_GROUP_MA=_F (NOM='pile_y2',
                                     UNION=pile_groups_y2),
                   MAILLAGE=mesh)

mesh = DEFI_GROUP (reuse=mesh,
                   CREA_GROUP_MA=_F (NOM='edge_s_',
                                     DIFFE=('edge_s', 'pile_x1', 'pile_x2',
                                           'pile_y1', 'pile_y2')),
                   MAILLAGE=mesh)

# Define calc. model
model =
AFFE_MODELE (AFFE=( _F (GROUP_MA=('cent_b_x', 'cent_b_y', 'edge_b_x1', 'edge_b_y1', 'edge_b_x2',
                                'edge_b_y2'),
                        MODELISATION=('POU_D_E', ),
                        PHENOMENE='MECANIQUE'),

```

```

_F (GROUP_MA=('cent_s_1','cent_s_2','cent_s_3','cent_s_4','edge_s_','corn_s','pile_x1','
pile_x2','pile_y1','pile_y2'),
MODELISATION=('DIS_T', ),
PHENOMENE='MECANIQUE')),
MAILLAGE=mesh)

# Define low-order elements
elemprop = AFFE_CARA_ELEM(DISCRET=(_F (CARA='K_T_D_L',
GROUP_MA=('cent_s_1', ),
VALE=(k_t, k_t, k_1)),
_F (CARA='K_T_D_L',
GROUP_MA=('cent_s_2', ),
VALE=(k_t, k_t, k_2)),
_F (CARA='K_T_D_L',
GROUP_MA=('cent_s_3', ),
VALE=(k_t, k_t, k_3)),
_F (CARA='K_T_D_L',
GROUP_MA=('cent_s_4', ),
VALE=(k_t, k_t, k_4)),
_F (CARA='K_T_D_L',
GROUP_MA=('edge_s_', ),
VALE=(k_t, k_t, k_5)),
_F (CARA='K_T_D_L',
GROUP_MA=('corn_s','pile_x1','pile_x2','pile_y1','pile_y2'),
VALE=(k_t, k_t, k_6))),
MODELE=model,
POUTRE=(_F (CARA=('A', 'IY', 'IZ', 'JX'),
GROUP_MA=('edge_b_x1', ),
SECTION='GENERALE',
VALE=(A_e_x1, I_e_x1, J_e_x1,
JT_e_x1)),
_F (CARA=('A', 'IY', 'IZ', 'JX'),
GROUP_MA=('edge_b_x2', ),
SECTION='GENERALE',
VALE=(A_e_x2, I_e_x2, J_e_x2,
JT_e_x2)),
_F (CARA=('A', 'IY', 'IZ', 'JX'),
GROUP_MA=('edge_b_y1', ),
SECTION='GENERALE',
VALE=(A_e_y1, I_e_y1, J_e_y1,
JT_e_y1)),
_F (CARA=('A', 'IY', 'IZ', 'JX'),
GROUP_MA=('edge_b_y2', ),

```

```

SECTION='GENERALE',
VALE=(A_e_y2, I_e_y2, J_e_y2,
JT_e_y2)),
_F(CARA=('A', 'IY', 'IZ', 'JX'),
GROUP_MA=('cent_b_x', 'cent_b_y'),
SECTION='GENERALE',
VALE=(A_rib, I_rib, J_rib,
JT_rib)))

# Define material properties

conc = DEFI_MATERIAU(ELAS=_F(E=15000000000.0,
NU=0.2,
RHO=2400.0))

soil_1 = DEFI_MATERIAU(DIS_CONTACT=_F(COULOMB=0.3,
DIST_1=0.5,
DIST_2=0.5,
RIGI_NOR=k_1,
RIGI_TAN=k_t))

soil_2 = DEFI_MATERIAU(DIS_CONTACT=_F(COULOMB=0.3,
DIST_1=0.5,
DIST_2=0.5,
RIGI_NOR=k_2,
RIGI_TAN=k_t))

soil_3 = DEFI_MATERIAU(DIS_CONTACT=_F(COULOMB=0.3,
DIST_1=0.5,
DIST_2=0.5,
RIGI_NOR=k_3,
RIGI_TAN=k_t))

soil_4 = DEFI_MATERIAU(DIS_CONTACT=_F(COULOMB=0.3,
DIST_1=0.5,
DIST_2=0.5,
RIGI_NOR=k_4,
RIGI_TAN=k_t))

soil_5 = DEFI_MATERIAU(DIS_CONTACT=_F(COULOMB=0.3,
DIST_1=0.5,
DIST_2=0.5,
RIGI_NOR=k_5,
RIGI_TAN=k_t))

```

```

pile_m = DEFI_MATERIAU (DIS_CONTACT=_F (COULOMB=1.0,
                                         DIST_1=0.5,
                                         DIST_2=0.5,
                                         RIGI_NOR=k_6,
                                         RIGI_TAN=k_t))

fieldmat
AFFE_MATERIAU (AFFE=_F (GROUP_MA= ('cent_b_x', 'cent_b_y', 'edge_b_x1', 'edge_b_y1', 'edge_b_x2', 'edge_b_y2'),
                        MATER=(conc, )),
              _F (GROUP_MA= ('cent_s_1', ),
                  MATER=(soil_1, )),
              _F (GROUP_MA= ('cent_s_2', ),
                  MATER=(soil_2, )),
              _F (GROUP_MA= ('cent_s_3', ),
                  MATER=(soil_3, )),
              _F (GROUP_MA= ('cent_s_4', ),
                  MATER=(soil_4, )),
              _F (GROUP_MA= ('edge_s_', ),
                  MATER=(soil_5, )),
              _F (GROUP_MA= ('corn_s', 'pile_x1', 'pile_x2', 'pile_y1', 'pile_y2'),
                  MATER=(pile_m, )),
              MODELE=model)

def mound_f(x,y):
# (omissis)
# Define functions

mound = FORMULE (NOM_PARA= ('X', 'Y'),
                 VALE_C='mound_f(X,Y)',
                 mound_f=mound_f)

listr = DEFI_LIST_REEL (DEBUT=0.0,
                       INTERVALLE=_F (JUSQU_A=1.0,
                                       NOMBRE=1))

times = DEFI_LIST_INST (DEFI_LIST=_F (LIST_INST=listr,
                                       NB_PAS_MAXI=10),
                       METHODE='AUTO')

fmult = DEFI_FONCTION (NOM_PARA='INST',
                      VALE=(0.0, 0.0, 1.0, -0.001))

```

```

# Define boundary conditions

fix_xy = AFFE_CHAR_MECA (DDL_IMPO=_F (DX=0.0,
                                         DY=0.0,
                                         GROUP_NO=('base_n', )),
                          MODELE=model)

ribs_l = AFFE_CHAR_MECA (FORCE_POUTRE=( _F (FZ=f_e_x_1,
                                             GROUP_MA=('edge_b_x1', )),
                          _F (FZ=f_e_y_1,
                              GROUP_MA=('edge_b_y1', )),
                          _F (FZ=f_e_x_2,
                              GROUP_MA=('edge_b_x2', )),
                          _F (FZ=f_e_y_2,
                              GROUP_MA=('edge_b_y2', )),
                          _F (FZ=q_,
                              GROUP_MA=('cent_b_x', 'cent_b_y'))),
                          MODELE=model)

mound_l = AFFE_CHAR_MECA_F (DDL_IMPO=_F (DZ=mound,
                                           GROUP_NO=('base_n', )),
                             MODELE=model)

# Perform non linear static analysis
SNL = STAT_NON_LINE (CARA_ELEM=elemprop,
                    CHAM_MATER=fieldmat,
                    COMPORTEMENT=( _F (GROUP_MA=('cent_b_x', 'cent_b_y', 'edge_b_x1', 'edge_b_y1', 'edge_b_x2', 'edge_b_y2'),
                                         RELATION='ELAS'),
                    _F (GROUP_MA=('cent_s_1', 'cent_s_2', 'cent_s_3', 'cent_s_4', 'edge_s', 'corn_s', 'pile_x1', 'pile_x2', 'pile_y1', 'pile_y2'),
                         RELATION='DIS_CHOC')),
                    CONVERGENCE=_F (ITER_GLOB_MAXI=10,
                                       RESI_GLOB_RELA=1e-05),
                    EXCIT=( _F (CHARGE=fix_xy),
                            _F (CHARGE=ribs_l),
                            _F (CHARGE=mound_l,
                                FONC_MULT=fmult)),
                    INCREMENT=_F (LIST_INST=times),
                    MODELE=model,
                    NEWTON=_F (REAC_ITER=1),
                    SOLVEUR=_F (METHODE='MULT_FRONT'))

```

```

# Calculate auxiliary fields

aux_1 = CALC_CHAMP (CONTRAINTE= ('SIPO_ELNO', ),
                    GROUP_MA= ('cent_b_x', ),
                    RESULTAT=SNL)

aux_2 = CALC_CHAMP (CONTRAINTE= ('SIPO_ELNO', ),
                    GROUP_MA= ('cent_b_y', ),
                    RESULTAT=SNL)

aux_3 = CALC_CHAMP (CONTRAINTE= ('SIPO_ELNO', ),
                    GROUP_MA= ('edge_b_x1', 'edge_b_x2'),
                    RESULTAT=SNL)

aux_4 = CALC_CHAMP (CONTRAINTE= ('SIPO_ELNO', ),
                    GROUP_MA= ('edge_b_y1', 'edge_b_y2'),
                    RESULTAT=SNL)

```

In this code snippet we report a partial version of a .comm file used to perform the FEA cloud-based calculation on a general rectangular stiffened slab on swelling soil. It is noted that some Python code is used to expand the capabilities of the Code Aster built-in command language and to include the necessary automation.

The partial results, stored temporarily in the cloud run filesystem, are further elaborated to produce the outputs for the frontend service. To perform this particular task, the FLASK Python framework was used as follows:

```

app=Flask(__name__)

@app.route('/api/ce', methods=['POST'])
def api_ce():

    try:

        input_data=request.get_json()

# (omissis)

# Return JSON reponse & status
        return
 jsonify({'max_values':max_values.to_json(), 'Dzex':Dzex_dict.to_json(), 'Dzey':Dzey_dict.
to_json(), 'Dzcx':Dzcx_dict_s.to_json(), 'Dzcy':Dzcy_dict_s.to_json(), 'Mcx':M_c_x_dict_f.
to_json(), 'Vcx':V_c_x_dict_f.to_json(), 'Mcy':M_c_y_dict_f.to_json(), 'Vcy':V_c_y_dict_f.

```



```

to_json(),'Mex':M_e_x_dict_f.to_json(),'Vex':V_e_x_dict_f.to_json(),'Mey':M_e_y_dict_f.
to_json(),'Vey':V_e_y_dict_f.to_json(),'3d_graph':html_graph}}, 200

    except Exception as e:

        return f"Exception: {e}"

# (omissis)

```

A2.6 Usage of CircuitPython

CircuitPython is an extremely versatile Python framework used to manage microcontrollers and sensor boards from numerous popular manufacturers.

In most of the lab scale tests single or coupled Adafruit MMA8451 accelerometers were used. Such sensors are accessed and controlled via the dedicated module through the I2C protocol. The following code snippet illustrates the use of CircuitPython to control a single accelerometer on a Raspberry Pi microcomputer.

```

try:
    # Initialize I2C
    i2c=busio.I2C(board.SCL,board.SDA)
    time.sleep(5.0)

    # Initialize MMA8451 module(s)
    sensor_A=adafruit_mma8451.MMA8451(i2c, address=0x1D)
    time.sleep(5.0)
    # Extra sensors here

except:
    # Exit with status 1
    raise Exception('Sensor(s) are offline')

# Acquisition parameters
freq=100.0
T_max=60.0
dT=1/freq
n_loops=math.floor(T_max*freq)
Continuous_mode=True
Angular_Threshold=20.0

# Acquire acceleration
array_A=np.empty((n_loops,3))

```

```

for j in range(0,n_loops):

    array_A[j,:]=np.array(sensor_A.acceleration)
    time.sleep(dT)

```

A2.7 Edge computing with Scipy and Scikit-Learn

The proposed edge computing framework involves basic digital signal processing founded on the Scipy python library and data-reduction based on the Scikit-Learn python library. The following code snippet is an example of one of such framework components:

```

# CALCULATE FEATURES
# Tilt w. low-pass filter
alpha = 0.5
f_acc_A=np.zeros(3)
roll_A=np.empty(n_loops)
pitch_A=np.empty(n_loops)

for j in range(0,n_loops):

    f_acc_A[:]=array_A[j,:]*alpha+f_acc_A[:]*(1-alpha)

    roll_A[j]=math.degrees(math.atan(f_acc_A[1]/math.sqrt(f_acc_A[0]**2+f_acc_A[2]**2)))
    pitch_A[j]=math.degrees(math.atan(-f_acc_A[0]/f_acc_A[2]))

# Peaks w. low pass filter
peak_acc_A=np.amax(f_acc_A, axis=0)

# EVALUATE STATUS
# Tilt Alarm
if (np.mean(roll_A) > Angular_Threshold) or (np.mean(pitch_A) > Angular_Threshold):
    Status='TILT ALARM'
else:
    Status='OK'

# Additional features in pandas DF (if Continuous mode ON)
if Continuous_mode==True:

    # Power spectrum w. low-pass filter
    f_x_p_A, Pxx_A = signal.welch(array_A[:,0], 1.0e3, nperseg=1024)
    f_y_p_A, Pyy_A = signal.welch(array_A[:,1], 1.0e3, nperseg=1024)
    f_z_p_A, Pzz_A = signal.welch(array_A[:,2], 1.0e3, nperseg=1024)

    # PCA on power spectra components

```

```

pca=PCA(n_components=3)
X=np.concatenate((Pxx_A, Pyy_A, Pzz_A), axis=0).reshape(-1,3)
pca.fit(X)

# Prepare pandas DB for export
dataset=pd.DataFrame({'Power_x_A' : Pxx_A, 'Power_y_A' : Pyy_A, 'Power_z_A' : Pzz_A})
print(dataset)

dataset_r=pd.DataFrame({'PCA_variance_ratio'      :   pca.explained_variance_ratio_,
'PCA_mean' : pca.mean_})

```

The above code is embedded in the Raspberry Pi microcomputer and allows for two basic edge computing functions:

1. In standard (low-consumption) mode: the sensing node elaborates a discrete time-series and sends out its status ('OK' or 'Tilt Alarm'). Acquisition and status reporting can be scheduled to balance power and network consumption and safety.
2. In continuous mode: the sensing node performs PSD evaluation and PCA reduction and sends out the results in full. Windowing can be chosen to allow for sound PSD and PCA execution.

Another framework component is a simple clustering algorithm used for unsupervised status classification. This time classification is performed on the data in full and it not constrained to any predefined criterion (e.g. Tilt as for the previous example):

```

from sklearn.cluster import MiniBatchKMeans
import numpy as np

# fit on PSD data
kmeans = MiniBatchKMeans(n_clusters=2,
                        random_state=0,
                        batch_size=10,
                        max_iter=10).fit(X)
# then use kmeans.labels_ to convey the clustering for further elaboration

```

KMeans (and especially mini-batch KMeans implementation) is a very quick way to identify general patterns in the data without the risk of overloading the edge computing node. In this example `n_clusters=2` was used, meaning that one is looking for a binary status just like in the 'OK' or 'Tilt Alarm' example above. A higher cluster number may be beneficial for certain applications (e.g. predictive maintenance), though it has not been tested at present.

A2.8 Cloud computing with Scikit-Learn

Two machine learning procedures were developed to be potentially implemented as cloud computing layers for the smart geocell. Such procedures are conceived to work on data-series from a large number of sensors.

The first procedure targets a simple supervised classification problem based on three status outputs (OK, pre-alarm and alarm). The core algorithm chosen to serve this purpose is Random Forests (RF) and its form is strongly based upon scikit-learn RF ensemble methods group. The simplest implementation for the prototype is reported in the following:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=1000, n_features=15, n_informative=3,
n_redundant=0, shuffle=False)
clf = RandomForestClassifier(n_estimators=10, n_outputs=3, max_depth=None,
min_samples_split=2, random_state=0)
clf.fit(X, y)
```

This implementation of the RF classifier involves the use of `sklearn.datasets.make_classification`. This command generates a random n-class classification problem, performing the necessary steps to mitigate any potential RF overfitting. This initially creates clusters of points normally distributed about vertices of an `n_informative`-dimensional hypercube with sides of length `2 x class_sep` (default `class_sep=1.0`) and assigns an equal number of clusters to each class. It introduces interdependence between these features and adds various types of further noise to the data. Without shuffling, X horizontally stacks features in the following order: the primary `n_informative` features, followed by `n_redundant` linear combinations of the informative features, followed by `n_repeated` (default `n_repeated=0`) duplicates, drawn randomly with replacement from the informative and redundant features. The remaining features are filled with random noise.

As for basic featuring, the edge computing procedures presented in par. A2.6 were used with the PSD+PCA pipeline logic. Since the XYZ power spectrum (or the relevant component of the spectrum after PCA) can be arbitrarily decomposed to determine the signal total power (i.e. variance) for specific frequency ranges, it is straightforward to create feature sets to train the RF algorithm. Since the first tests were based solely on very limited time series acquired in controlled lab conditions, further experimentation is needed to balance the featuring cost with the RF classifier prediction accuracy.

The second procedure focuses on regression, with the goal of deep data regularization for integrated predictive maintenance purposes. The core supervised algorithm chosen to perform this task is ElasticNet and its form is strongly based upon scikit-learn linear models group. The simplest implementation for our prototype is reported in the following:

```
from sklearn.linear_model import ElasticNet
from sklearn.datasets import make_regression

X, y = make_regression(n_features=9, random_state=0)
regr = ElasticNet(random_state=0)
regr.fit(X, y)
```

This implementation of Elastic Net involves the use of `sklearn.datasets.make_regression`.

This command generates a random regression problem. The input set can either be well conditioned (by default) or have a low rank fat-tail singular profile. The output is generated by applying a (potentially biased) random linear regression model with `n_informative` (default `n_informative=10`) nonzero regressors to the previously generated input and some gaussian centered noise with adjustable scale.

As for the previous RF implementation, the PSD+PCA pipeline was used for featuring. The simple output descriptors obtained by fitting the Elastic Net regressor have the potential to be used in a broader, multi-sensors predictive maintenance monitoring system, targeting e.g. time-series trends

A2.9 Blockchain as a data security layer

The proposed blockchain implementation involves the use of the Hyperledger Iroha Python library from the Hyperledger Iroha Project. In the blockchain assets are represented by data pre-elaborated on the edge, e.g. PSD batches. High-permission nodes (granted with the `can_create_asset` permission) might be identified with single sensors or gateways, being the entities that formally create the asset (`PSD_batch_hash`). Low-permission nodes, on the other hand, might be identified with e.g. cloud storage nodes, cloud computing nodes, etc.

The main purpose of this (very simplistic) implementation is to interpose an additional security layer in between the acquisition endpoints and the data mining process that entails the distinctive features of the private blockchains. This security layer is basically an immutable, cryptographic ledger.

In the example reported in the first snippet below, the admin creates a domain that contains the `can_create_asset` permission and only one user (`sensor_1`). In this example `sensor_1` can create the asset (i.e. `PSD_batch_hash`).

```

admin = commons.new_user('admin@test')
sensor_1 = commons.new_user('sensor_1@test')
iroha = irohalib.Iroha(admin['id'])

@commons.hex
def genesis_tx():
    test_permissions = [primitive_pb2.can_create_asset]
    genesis_commands = commons.genesis_block(admin, sensor_1, test_permissions)
    tx = iroha.transaction(genesis_commands)
    irohalib.IrohaCrypto.sign_transaction(tx, admin['key'])
    return tx

@commons.hex
def create_asset_tx():
    tx = iroha.transaction([
        iroha.command('CreateAsset', asset_name='PSD_batch_hash', domain_id='test',
precision=0)
    ], creator_account=sensor_1['id'])
    irohalib.IrohaCrypto.sign_transaction(tx, sensor_1['key'])
    return tx

```

Similarly, one can conceive a more complex ledger that involves, as anticipated high-permission as well as low-permission nodes:

```

@commons.hex
def genesis_tx():
    test_permissions = [primitive_pb2.can_transfer, primitive_pb2.can_receive]
    genesis_commands = commons.genesis_block(admin, sensor_1, test_permissions)
    genesis_commands.extend([
        iroha.command('CreateAccount', account_name='cloud_1', domain_id='test',
public_key=irohalib.IrohaCrypto.derive_public_key(cloud_1['key'])),
        iroha.command('CreateAsset', asset_name='PSD_batch_hash', domain_id='test',
precision=0),
        iroha.command('AddAssetQuantity', asset_id='PSD_batch_hash#test',
amount='1000'),
        iroha.command('TransferAsset',
                        src_account_id=admin['id'],
                        dest_account_id=sensor_1['id'],
                        asset_id='PSD_batch_hash#test',
                        description='init top up',
                        amount='1000')
    ])

```

```

    ])
    tx = iroha.transaction(genesis_commands)
    irohalib.IrohaCrypto.sign_transaction(tx, admin['key'])
    return tx

@commons.hex
def transfer_asset_tx():
    tx = iroha.transaction([
        iroha.command('TransferAsset',
            src_account_id=sensor_1['id'],
            dest_account_id=cloud_1['id'],
            asset_id='PSD_batch_hash#test',
            description='transfer to Cloud node 1',
            amount='100')
    ], creator_account=sensor_1['id'])
    irohalib.IrohaCrypto.sign_transaction(tx, sensor_1['key'])
    return tx

```

In the above example admin generates a dummy asset and transfers it to the sensor_1 node. The permission schema allows then sensor_1 to transfer assets to cloud_1 and cloud_1 is granted with the permission to receive such asset. The ledger will then be able to record the transaction permanently, storing the unique hash of the data batch that is being transferred through the network. More complex rules (e.g. transaction limits) can be implemented in the code.

It is noted that blockchains are not a convenient way to store massive data. Hence, as anticipated, the proposed blockchain implementation is based solely on data hashes transactions. Bulk data storage, at least in the current prototype implementation, is thought to be performed classically through dedicated proprietary servers and/or cloud storage services (which are identified as low-permission nodes/users of the private blockchain).