DEVELOPMENT OF GIANT MAGNETORESISTIVE BIOSENSORS AND SYSTEMS FOR EARLY DISEASE DETECTION

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE
UNIVERSITY OF MINNESOTA
BY

TODD KLEIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF ELECTRICAL ENGINEERING

PROFESSOR JIAN-PING WANG

FEBRUARY 2018

Acknowledgements

- Thanks to the Chorzempa family for your generous donation to support this work.
- To Professor Jian-Ping Wang: Thank you for your steady guidance, patience, and understanding. It means more to me than you know.
- To Professor Amy Skubitz, Kristin Boylan, Rachel Isaksson Vogel, and Dr. Levi Downs, M.D.: Thanks for your valuable collaboration on the Ovarian Cancer Translational Pilot Award through the Department of Defense.
- To all my group members: Thank you for the tremendous teamwork and unwavering support. I have depended on you so much.
- To my committee members: Thank you for the valuable insights that helped steer me towards practical solutions in experimental design.
- To the staff at the University of Minnesota Nanofabrication Center, Institute of Rock Magnetism, and Characterization Facility: Thank you for providing such a high level of support. It has truly been a joy to work with you.
- To the Golden Gopher Magnetic Biosensing XCHALLENGE team members: Thank you all for your hard work and trust that allowed us to attain the hand-held GMR biosensing system.
- This work was supported in part by the U.S. Department of Defense under the Ovarian Cancer Translational Pilot Award W81XWH-11-1-0496. In addition, we acknowledge the generous support from the National Science Foundation under award number BME 0730825, and the NINN Research Experience for Undergraduates Program. Parts of this work were carried out in the University of Minnesota I.T. Characterization Facility, which receives partial support from NSF

through the NINN program. Parts of this work were also supported by the Institute of Engineering in Medicine and University of Minnesota Supercomputing Institute.

- GMR sensors for the work in chapter 7 are similar in construction to the ones described in chapter 2 but were processed and assembled by Zepto Life Technology, a company founded by Professor Jian-Ping Wang of the Department of Electrical and Computer Engineering at the University of Minnesota.

Dedication

To my wife Sara: You are the only other person who truly understands the level of dedication required to finish this work. Thank you for your love and patience.

To my son Evan: Fatherhood is the greatest experience I have ever known.

Abstract

Giant magnetoresistance (GMR) biosensors have been used with great success for the detection of a variety of biomarkers. Linear GMR biosensors were first proposed in 1998 and have been used since then. The major scientific contribution of this dissertation is to go beyond the previously published models for the linear GMR biosensor and provide analysis of practical design decisions that occur during sensor fabrication. I highlight the central role played by GMR free layer stray field near the sensor edge and present a numerical calculation to guide future linear GMR biosensor design. In addition, I explain how I proposed and demonstrated a novel, non-linear, domain-wall based GMR biosensor, in an effort to detect single molecules. I also describe my contributions to an effort to understand and build upon a previous experimental result using a large area sensor with multidomain switching.

The major technological contribution described in this dissertation is the development of a GMR biosensing system that can potentially contribute to the early detection of ovarian cancer and serve as a platform for detecting a wide variety of other biomarkers. System integration included spintronic and nanomagnetic materials engineering, design of a coil with a ferrite core, electrical engineering, analog and digital signal processing, firmware programming, user interface programming on both a PC and an Android smartphone, communications over both USB and Bluetooth, and mechanical design.

Hand-held and bench-top systems of this GMR bionsensor were both developed. Both versions use the same sensors, electrical hardware, firmware, and software, but differ mechanically and in the number of sensors available per assay. The bench-top version was

completed first and used to demonstrate high sensitivity multiplex detection of ovarian cancer biomarkers CA-125, HE-4, and IL-6, with limits of detection below 10 pg/mL. The hand-held version was then completed and used with a preliminary biotin-streptavidin demonstration. Further development of the hand-held system involves integrating microfluidics.

Table of Contents

List of Tables

List of Figures

List of Equations

List of Abbreviations

AFM	Atomic force microscopy
ALD	Atomic layer deposition
AMR	Anisotropic magnetoresistance
APTES	3-aminopropyltriethoxy silane
CA-125	Cancer antigen 125
CDC	Centers for Disease Control
CIP	Current-in-plane
CLIA	Clinical Laboratory Improvement Amendments
CMOS	Complementary metal oxide semiconductor
CPP	Current-perpendicular-to-plane
CRP	C-reactive protein
DOS	Density of states
DW	Domain wall
ELISA	Enzyme-linked immunoabsorbent assay
FDA	Food and Drug Administration
Glu	Glutaraldehyde
GMR	Giant magnetoresistance
HE-4	Human epididymis protein 4
IL-6	Interleukin 6
LFIA	Lateral flow immunochromatographic assay
MFM	Magnetic force microscopy
MNP	Magnetic nanoparticle
NHS	*N*-Hydroxysuccinimide
NFC	Nanofabrication Center (UMN)
PBS	Phosphate buffered saline
PBST	Phosphate buffered saline with Tween20
PCB	Printed circuit board
RKKY	Ruderman–Kittel–Kasuya–Yosida
ROMA	Risk of ovarian malignancy algorithm
SEM	Scanning electron microscope
TMR	Tunneling magnetoresistance
VSM	Vibrating sample magnetometer

1. Introduction

1. Motivation: The need for new biosensing technology

Technologies that facilitate early detection of disease provide a path toward less expensive, more effective, and sustainable healthcare. The motivation behind this dissertation was to develop such a technology. More specifically, our research aimed to develop a portable GMR biosensing system that could contribute to the early detection of ovarian cancer and serve as a platform for detecting a wide variety of other diseases. We will begin by reviewing the needs and challenges related to early detection of ovarian cancer before returning to the description of GMR biosensing technology.

2. Early detection of ovarian cancer

Although ovarian cancer is not the most common cancer, it is particularly insidious due to the lack of recognizable early symptoms. As of 2016 in the United States, 22,280 women are diagnosed with a new case of ovarian cancer, and more than 14,240 women die from ovarian cancer [1]. Ovarian cancer is the fourth most common cause of female cancer death in the developed world due to its late stage at the time of disease diagnosis [2]. Since treatment works best when ovarian cancer is found in its early stages, we know that early detection is key to survival.

SEER has compiled statistics for the five-year survival rate versus diagnosis stage. Their data from 2007-2013 in the United States, shown in **Fig. 1.1**, highlights the importance of early diagnosis [3]. A biosensing system that can rapidly and inexpensively detect ovarian cancer would, then, be a huge advance in the battle against this disease.

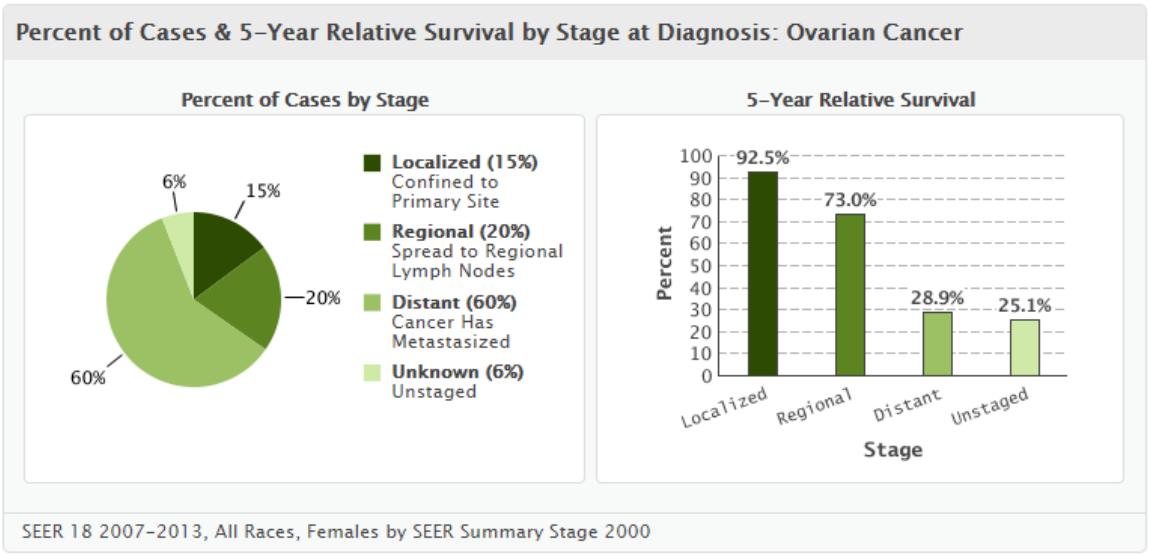

Fig. 1.1. Ovarian cancer statistics from 2007-2013 in the United States. A) Stage of ovarian cancer at time of diagnosis. B) 5-year survival rate versus ovarian cancer stage at time of diagnosis. [3]

2.1. What is needed

The best known individual protein biomarker with correlation to ovarian cancer is CA-125. However, the utility of this single biomarker protein is limited. CA-125 levels have been found to be elevated in 80% of patients with late stage ovarian cancers, but less than half of those with early stage ovarian cancer [4, 5]. To be useful as a screening tool, a positive predictive value greater than 10% is required [5]. According to SEER data from 2014, the number of new ovarian cancer cases was 11.7 per 100,000 women. Therefore, even with 100% sensitivity, we require specificity of 99.9%. To date, no assay has been FDA approved for screening for ovarian cancer in the general population.

Improvements in sensitivity and specificity have been offered by second-generation CA-125 II along with multi-analyte assays. The ROMA algorithm combines concentrations of both CA-125 II and HE4 [6, 7]. This represents an advance over just CA-125 alone, but still leaves much room for improvement. For this reason, ROMA has been

approved by FDA only to track recurrence of known ovarian cancer but not to screen for new cases. Vermillion, Inc. has completed two generations of ovarian cancer multivariate index assays (OVA1 and Overa) [8, 9]. The second-generation assay (Overa) combines concentrations of A-1, CA-125 II, HE4, FSH, and TRF. Sensitivities and specificities of ovarian cancer assays are listed in Table 1.1 below [7].

Table 1.1. Test performance for detecting ovarian cancer of all histologic types. [7]

Biomarker	Sensitivity	Specificity
CA125 *,+,#	76%	94%
Ova1 *	94%	54%
ROMA ^	89%	83%
Overa *	91%	69%

* Studied in same patient population; + CA125-II assay (second generation); # CA125 not FDA-approved for preoperative use; ^ Meta-analysis [10]

Even with such improvements, ovarian cancer biomarker assays have not yet been approved by FDA for early detection by screening. It is possible that we will need more than five biomarkers, and research into new biomarkers is ongoing. For this work, we note the practical importance of devices that perform multiplex biomarker assays. This is a central theme to this dissertation, and Chapter 8 describes the multiplex detection of three ovarian cancer biomarkers.

In addition to the importance of multiplex assays, we must also note that ovarian cancer *in vitro* diagnostic tools are not widely available. Mayo Clinic and Quest Diagnostics both provide such tests for ROMA (CA-125 and HE-4). Quest does not publicize what equipment they use for testing, but Mayo Clinic uses the Roche Cobas 6000 for ROMA ovarian cancer diagnostics. Point-of-care systems, if made inexpensive and widely distributed, could help reverse this limitation. Once an FDA-approved screening assay is

developed, we will need to reduce the cost, effort, and required expertise associated with the procedure to increase access to ovarian cancer screening.

Ideally, point-of-care settings would have a device that is small, affordable, fully automated, and with the speed, sensitivity and specificity of central laboratory equipment. As already noted, this system must also be able to run a multiplex assay so that at least 3-5 ovarian cancer protein biomarkers could be detected simultaneously. We will consider each of these qualities in turn. Our focus here is on the physical sensing system rather than any particular set of ovarian cancer biomarkers.

2.1.1. Automated, affordable, and portable

If the system is automated, affordable, and portable, then it can be distributed to many clinics across the world and provide the largest possible impact. In the case of the United States, it is also important that the system be CLIA-waived (based on the Clinical Laboratory Improvement Amendments of 1988), which means that it does not require knowledge and skills specific to central laboratory technicians. For the CLIA waiver, a system would require automation so that whole blood is processed as required by the assay technology. Generally, this means that whole blood is filtered so that only serum is left before mixing with assay reagents and moving to the sensing area. This requirement can be met by systems that include a microfluidics cartridge.

2.1.2. Speed, sensitivity and specificity comparable to a central lab system

For a system to produce near perfect assay sensitivity and specificity, the instrument's sensitivity and precision need to match that of central laboratory instruments. Currently, no point-of-care instruments on the market can reach this level of performance. Lessons learned from the development of ROMA and Overa assays teach us that any sacrifice is sensitivity and specificity cannot be expected to achieve FDA approval.

Of course, if we wish to increase the number of patients that are screened, we need to provide this sensitivity and specificity in a short enough time to match the required throughput. For the system to screen as many patients as possible, it needs to be fast and efficient so that it does not take up valuable (and likely costly) time of lab employees. As a reasonable target, the system would deliver a result in under 30 minutes. This is particularly challenging in the case of multiplex biomarker assays. Since it is reasonable to assume that each individual biomarker assay would each require 15 to 30 minutes, all assays should run in parallel.

2.1.3. Multiplex

Focus on multiplex assays are critical to the development of a device with FDA approval for screening ovarian cancer. While developing an instrument for ovarian cancer screening, we do not know in advance which biomarkers will be selected for the screening assay since none have reached the sensitivity and specificity demanded by FDA. Based on the history of ROMA and Overa assays, we can safely assume that at least 3 biomarkers will be needed, and it is likely that the number will exceed 5 biomarkers to achieve high enough sensitivity and selectivity for FDA approval as a screening tool.

GMR-based magnetic biosensing is particularly well suited for multiplex assays. Not only can many GMR sensors fit into a small area, but the magnetic nanoparticles (MNPs) have short-range interaction with sensors due to the $1/d^3$ reduction in stray field with increased distance (d) between sensor and MNP. This means that the MNPs that serve as assay quantification labels for one sensor do not interfere with assay quantification on neighboring sensors within the same sample volume. For this reason, we conclude that GMR biosensors are particularly well suited for multiplex assays.

2.2. Overview of available technologies

Many alternative technologies are competing to produce a device that is small, affordable, fully automated, and with the sensitivity and precision of central laboratory equipment while delivering multiplex results. We have categorized the technologies to focus solely on the mode of sensing: Optical, Electrochemical, Field-effect, and GMR. Each of these technologies has its strengths and weaknesses. We must note that microfluidic technology does not constitute a unique sensing mode as it is quite general and can be integrated into many different biosensing systems.

Table 1.2. Available technologies for ovarian cancer detection

Alternative Technologies			
Technology	Limit of Detection	Multiplex	Reference
Optical			
Triage "Ascend" (LFIA)	3×10^{-7} mol/L	7	[11]
Pillar chip (Fluorescent)	5 pg/mL	2	[12]
Optical waveguide (Monolithic)	2×10^{-14} mol/L	3	[13]
Electrochemical			
Gold nanoparticles / Carbon nanospheres	0.0014 U/mL	1	[14]
Carbon nanofiber array	20 ng/mL	3	[15]
Field-Effect			
Nanowire	5×10^{-16} mol/L	3	[16]
Graphene transistor	1×10^{-9} mol/L	2	[17]
Gold coated carbon nanotube array	50 pg/mL	1	[18]

GMR			
Large area GMR	4×10^{-12} mol/L	1	[19]
Linear GMR	1 pg/mL	3	[20]
Linear GMR	1 ng/mL	3	[21]

2.2.1. Optical

Optical biosensing technologies tend to be the most common. Fluorescent technologies often depend on lasers, lenses, and photomultipliers and are therefore vulnerable to system drift and difficult to miniaturize. The LFIA, on the other hand, is portable, very inexpensive, easy to use, and even multiplexed in some cases [11]. One of the most successful and familiar examples of LFIA technology is the in-home pregnancy test. LFIA struggles with sensitivity and precision but is a good candidate for certain applications that have a very wide distinction between pass and fail. That is clearly not the case with ovarian cancer.

A newer diffusion driven technology with optical sensing is the Pillar Chip [12]. Like LFIA, it uses capillary action to move fluids. To increase sensitivity, the reader is a fluorescent scanner that can scan one cartridge at a time, with a limit of detection of 5 pg/mL for NTproBNP [12]. Also, it was able to perform a multiplex assay with NTproBNP plus CRP with a coefficient of variation 5-15%. This performance makes the Pillar Chip a very good competitor, but fluorescent scanners need to deal with sensor drift and recalibration issues.

To avoid the issues introduced by fluorescent immunoassays, researchers have extensively explored the optical waveguide as an option for point-of-care settings. One notable example includes silver-plated gold nanoparticle labels [13]. This sensor was able

to achieve a limit of detection of 20 fM while performing a multiplex assay detection of Streptavidin and anti-mouse IgG. The design is a monolithic silicon device with LED and photodetector on opposite sides of an optical fiber [13]. Optical waveguide biosensing has generally suffered from problems associated with equipment-generated noise [22], and it has only recently reached a proof-of-concept stage with non-biological molecules, with claims of good sensitivity and a low detection limit [22]. It remains to be seen if this technology can be developed enough to accommodate multiple biomarkers for rapid, multiplex detection.

2.2.2. Electrochemical

The Roche Cobas 6000 performs an electrochemiluminescence immunoassay to detect CA-125 and HE-4 in the ROMA assay [23]. This is a quite sophisticated extension of electrochemical sensing where Ruthenium atoms act as immunoassay labels that emit light due to redox reactions when voltage is applied to the system. The light from Ruthenium labels is then detected through a photomultiplier. Again, we note that such a system is quite effective but difficult to miniaturize and requires frequent calibration since photomultipliers are prone to drift.

Electrochemical immunoassays also derive their signal from redox reactions but can be miniaturized due to relative simplicity of hardware. In general, such systems apply a series of voltage pulses and measure the current resulting from both reversible and irreversible components of redox reactions. In some implementations, the assay depends on a label [14] and other implementations are label-free [15]. The main question for this technology is whether it can ever reach the level of sensitivity offered by electrochemiluminescence. Additionally, we must note the electrochemical immunoassays are not easy to multiplex since the signal is derived from electrical current flowing through the sample itself.

Therefore, research into developing multiplex assays is quite important for electrochemical immunoassay technologies [15].

2.2.3. Field-effect

The interface between solid surfaces and liquids is known to harbor an electric charge and consequent electric potential known as the zeta potential. This zeta potential can be either positive or negative, and it depends on the chemical composition of both the solid and the liquid. In the case of protein biomarkers, the zeta potential will be negative for the relevant pH levels within the aqueous environment of an immunoassay. The working principle of field-effect biosensors is to use this protein zeta potential to induce detectable change in a semiconductor device.

Field-effect sensors provide one approach to label-free immunoassays, and this is clearly an advantage in terms of simplicity and flexibility. It may be possible that such a sensor could perform affinity measurements in addition to *in vitro* diagnostics. The first field-effect devices were based on the silicon-based field-effect transistor, but more recent research focuses on silicon nanowires [16], graphene transistors [17], and gold-coated carbon nanotube arrays [18].

2.2.4. GMR biosensing and ovarian cancer detection

One technology that can meet all the requirements for ovarian cancer detection and which may have fewer weaknesses than the technologies noted above is GMR-based magnetic biosensing. Magnetic sensing and quantification of biological materials such as DNA [24, 25, 26, 27] and proteins [19, 28, 29, 30, 31, 32, 33, 34, 21] has become increasingly important due to the need for early detection of diseases. Since its first inception [35, 36, 37, 38, 39], one very promising platform [40, 41, 42, 43, 44] for this application is the use of GMR [45, 46, 47] elements sensitive to the presence of MNPs. It

has been shown that GMR-MNP sensors have the potential for ultra-sensitive, high-speed, multiplexed, inexpensive and portable biodetection [31, 32, 48, 49, 50, 51, 52, 53]. One of the fundamental advantages of this and other magnetic biosensing systems is the negligible background of magnetic material in biological samples. This leads to an inherent advantage for low-concentration detection [20, 53], and we can predict signals with an accurate sensor model [37, 54, 55]. Although previous literature establishes this model quite well, film deposition and fabrication details that dramatically impact the sensor performance and optimal operating conditions have not been well studied. For this reason, the research for this project began with studying simple variations in GMR film performance and fabrication configurations.

3. Overview of this dissertation

The dissertation describes two major contributions to the field of biosensing. Most notable is the design and integration of a GMR biosensing system along with the demonstration of its performance. System integration included spintronic and nanomagnetic materials engineering, design of coil with ferrite core, electrical engineering, analog and digital signal processing, firmware programming, user interface programming on both a PC and an Android smartphone, communications over both USB and Bluetooth, and mechanical design. A hand-held version and bench-top version were both developed (chapter 6). Both versions use the same sensors, electrical hardware, firmware, and software, but differ mechanically and in the number of sensors available per assay. Demonstration of the bench-top version of the GMR biosensing system in the context of ovarian cancer detection included two gold standard protein biomarkers (CA-125 and HE-4) plus a third (IL-6). This GMR biosensing system achieved multiplex detection of CA-125, HE-4, and IL-6 at a limit of detection less than 10 pg/mL (chapter 7).

The second major contribution to the field was a systematic description of the interaction between MNPs and the linear GMR biosensor [55]. The main point is that there is a strong dependence of signal strength on MNP location relative to the sensor edge, and this effects the sensor design. This effect was known in previous literature [56], but the impact on design was not systematically discussed until our work (chapter 3). In addition to these two major contributions, we developed a variety of GMR biosensors to understand and build upon a previous result using a large area sensor with multidomain switching (chapters 4 and 5). Fabrication techniques used to develop the biosensors are presented in chapter 2.

3.1. Variety of sensor designs

3.1.1. Large area multidomain switching sensor

Previous success with a large area sensor using high-moment CoFe MNPs [19, 34] made the large area sensor the intended design for this project. However, the newly deposited GMR films had much different switching characteristics than expected and was much better suited to a low-hysteresis "linear" sensor design. The linear sensor was already well established in the literature, so we felt comfortable pursuing this design. At the same time, we worked to develop a theoretical model for a large area sensor to explain the data from our previous work (see chapter 4) and learn how to build on it more effectively.

3.1.2. Linear sensor

The newly deposited GMR films were characterized by a uniform free layer with low hysteresis. This type of film is ideal for the linear sensor [35, 54, 55] and represents the most completely studied GMR biosensor design. Prior to fabrication, we updated our theoretical understanding of the technology and thereby optimized our design (see chapter

3). By choosing this relatively mature technology, we were better able to collaborate with end users while at the same time prepare for the future by researching new sensor designs.

3.1.3. Domain wall sensor

While working on the theoretical model for the large area multidomain switching sensor, we developed an analytical model for weak domain wall pinning by a MNP stray field [57] and then tested the performance of a GMR sensor based on domain wall pinning (see chapter 5) [58]. In doing so, we demonstrated the detection of just seven MNPs with 35 nm diameter each with a head-to-head domain wall within the free layer of a GMR spin valve. The detection consisted of an 8 Oe shift in the field required to dislodge (or "de-pin") the trapped domain wall from its natural pinning site within the device. This result was deemed statistically significant from 0 at a 0.05 significance level.

3.2. Multiplex detection of CA-125, HE-4, and IL-6

Our objective for system development was to quantify the levels of three ovarian cancer biomarkers – CA-125, HE-4, and IL-6 – in a portable and multiplex GMR biosensing system. Multiplexed dose-response curves for CA-125, HE-4, and IL-6 were completed with a limit of detection below 10 pg/mL (see chapter 7). This is a significant step in terms of hardware but does not yet provide an assay for early detection of ovarian cancer. Assay development with required sensitivity and specificity will be required, and we believe the GMR biosensing system will provide an excellent platform for such development.

4. Key concepts and terminology for GMR biosensing

Below are terms and concepts vital for understanding the three GMR biosensors described in this work. Most details in this section can also be found in magnetic material textbooks [59, 60] and spintronics literature [61, 62].

4.1. Magnetic energies

4.1.1. Zeeman energy

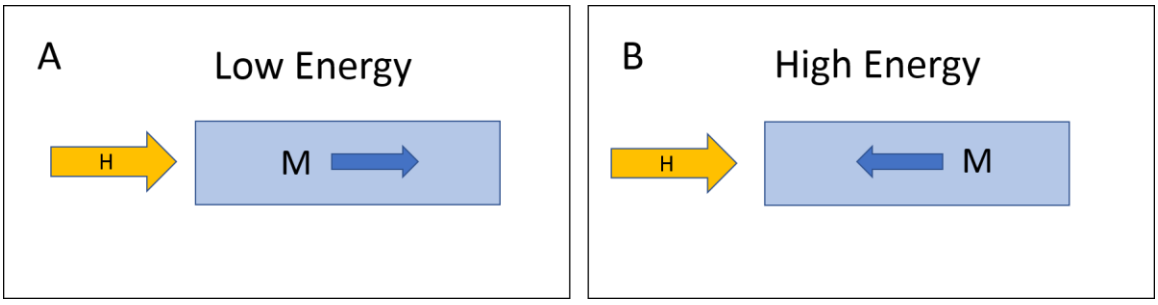

Fig. 1.2. Zeeman energy is minimized when magnetization aligns parallel to applied magnetic field as shown in A. It is maximized when magnetization aligns antiparallel to applied magnetic field, as shown in B.

Zeeman energy is the energy that causes magnetic switching to occur in the presence of an applied magnetic field. As illustrated in **Fig 1.2**, the low energy conditions occur when the magnetization direction of a magnetic material coincides with the direction of the applied field. Zeeman energy is calculated by the vector dot product of magnetization and external applied field: $E = -M \cdot H$ (cgs) or $E = -M \cdot B$ (SI). In the case of biosensing, this is the energy that couples the external field, particle, and sensor together.

4.1.2. Shape anisotropy

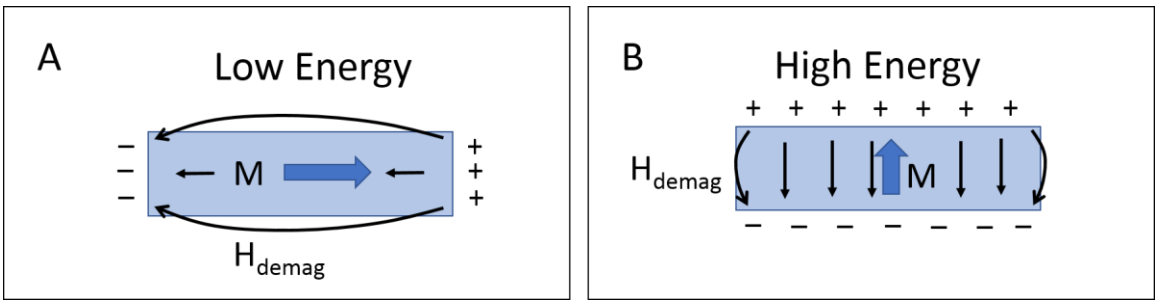

Fig. 1.3. An object magnetized along the long axis as shown in A has lower energy than the same object magnetized along the short axis as shown in B due to the Zeeman energy related to the demagnetizing field.

As illustrated in **Fig 1.3**, magnetization prefers to align itself along the longest axis of an object due to Zeeman energy minimization. This can be visualized in terms of the magnetic field produced by hypothetical uniform magnetization of a rectangular ferromagnetic bar. The total magnetic charge will be greatest in the case of magnetization along the shortest axis. At the same time, this configuration brings the interior material very close to the magnetic charge and hence very strong magnetic field. For this reason, it represents the highest Zeeman energy configuration. Conversely, magnetization along the longest axis produces relatively small total magnetic charge, and it is relatively far from the interior magnetic material and therefore represents low Zeeman energy. In the case of magnetic biosensing, the shape anisotropy is used to set the easy axis direction of the GMR spin valve free layer.

4.1.3. Crystal anisotropy

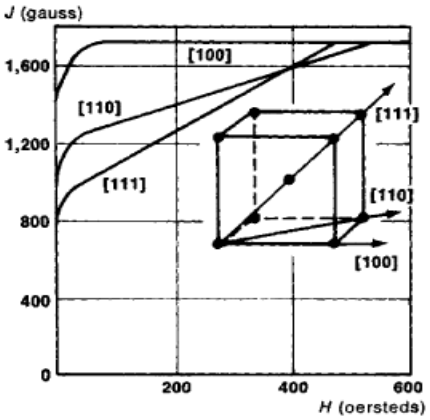

Fig. 1.4. Magnetization along various directions in a crystal lattice differ in energy due to spin-orbit coupling. [98]

Crystal anisotropy energy originates from the spin-orbit coupling of electrons. Since electron spin produces magnetization and electron orbits are tied to crystal orientation, the spin-orbit coupling energy ties preferred magnetization direction to crystal orientation as shown in **Fig. 1.4**. In some cases, this energy is greater than shape anisotropy. In the case of magnetic biosensing, we wish to minimize crystal anisotropy in the free layer of the GMR spin valve so that it can freely rotate in the presence of applied magnetic fields due to Zeeman energy.

4.1.4. Induced anisotropy

Induced anisotropy can occur in magnetic alloys (e.g. CoFe) while annealing under a magnetic field or during film deposition within a magnetic field. This occurs from pair ordering in a binary ferromagnetic alloy. The easy axis will align with the ordered pairs as illustrated in **Fig. 1.5** above. In the case of magnetic biosensing, induced anisotropy occurs during deposition of the GMR spin valve thin film because a magnetic field is applied in the plane of the wafer during deposition.

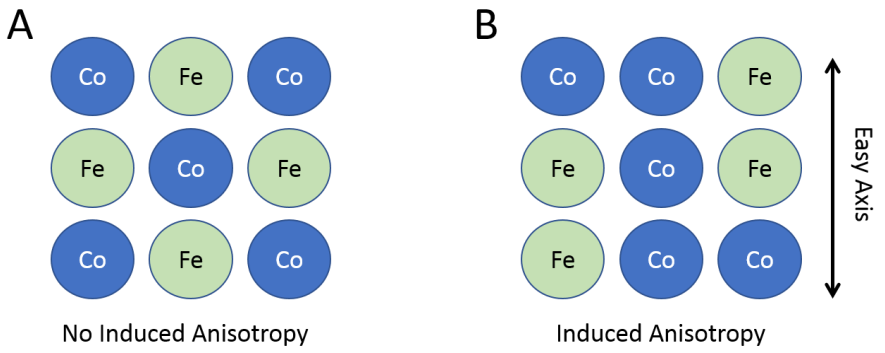

Fig. 1.5. Applied magnetic field during film deposition or magnetic annealing can induce pair ordering and consequent anisotropy in binary ferromagnetic alloys. This type of anisotropy is called "induced anisotropy".

4.1.5. Exchange coupling effect

Exchange energy is a quantum phenomenon that causes adjacent electrons to prefer either parallel or antiparallel orientation depending on the distance between the electrons. It is closely related to the familiar Fermionic character of electrons, which dictates that no two electrons can share quantum numbers and occupy the same space. Exchange energy is what causes ferromagnetism, antiferromagnetism, the interfacial coupling of adjacent

magnetic thin films, and even the indirect coupling of magnetic thin films mediated by low spin-scattering materials such as copper or ruthenium. **Fig 1.6** illustrates the both the indirect coupling phenomenon known as the RKKY interaction as well as exchange occurring at an interface.

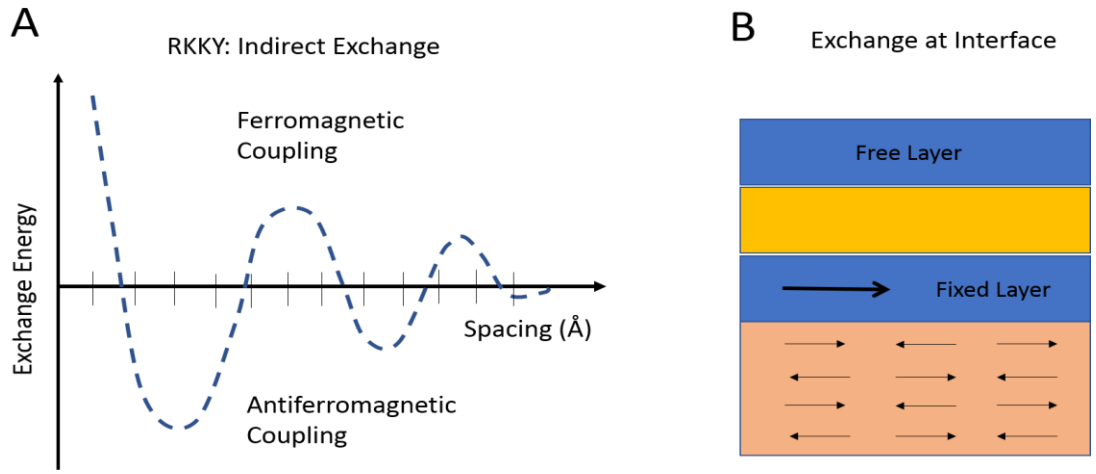

Fig. 1.6. Exchange energy is critical to the GMR biosensor. A) In systems with two ferromagnetic thin films separated by a polarizable non-magnetic metal (e.g. Ruthenium), we see the ferromagnetic films couple either parallel of antiparallel depending on the gap. B) In systems with an antiferromagnet / ferromagnetic interface, exchange coupling can cause the ferromagnetic layer magnetization direction to remain nominally fixed.

In the work presented here, we utilize the interfacial coupling between a ferromagnetic layer and antiferromagnetic layer that results from exchange energy. Since the antiferromagnetic layer has very little response at all to an applied magnetic field, the interfacial exchange causes the ferromagnetic layer to become fixed and hence serve as a reference layer in the GMR spin valve.

4.2 Micromagnetic simulation

All the magnetic energies listed above can be combined into a micromagnetic simulation where the microstructure of magnetic materials is modeled by individual cubes of material. The size of each cube is somewhat arbitrary but often in the range of 1-5nm on

each side. The equation of motion for each micromagnetic element is governed by the Landau-Lifshitz-Gilbert equation (**Eq. 1.1**), where the effective field H_{eff} is composed of the exchange field, the anisotropy field, applied field, and demagnetizing field. This equation is used by micromagnetic software such as OOMMF to model the microstructure of magnetic materials. We recently used OOMMF to model domain nucleation in the large area GMR biosensor [63].

$$\frac{\partial \boldsymbol{m}}{\partial t} = -|\gamma|\boldsymbol{m} \times \boldsymbol{H}_{eff} + \alpha \boldsymbol{m} \times \frac{\partial \boldsymbol{m}}{\partial t} \qquad \textbf{Eq. 1.1}$$

$$\boldsymbol{H}_{eff} = \frac{2A}{\mu_0 M_s}\nabla^2 m - \frac{1}{\mu_0 M_s}\frac{\partial F_{anis}}{\partial \boldsymbol{m}} + \boldsymbol{H}_a + \boldsymbol{H}_d$$

4.3. Magnetic reversal

4.3.1. Coherent rotation

In the case of uniform magnetization in a single crystal ellipsoid, we can consider magnetization to remain uniform even while an external field drives switching of the magnetization direction. The simplest case to consider is one in which there is a single easy axis, often dictated by shape anisotropy as occurs in the linear sensor design for magnetic biosensing. In this case, we can apply the Stoner-Wohlfarth model of coherent rotation which illustrates a prototypical family of hysteresis loops depending on the relative direction between the external applied field and easy axis.

From the Stoner-Wohlfarth model, we can define many useful characteristics of real magnetic switching. First, we note that the magnetization curves are called hysteresis loops due to their delayed switching from overcoming an energy barrier. The field required to obtain zero magnetization along the hysteresis loop is called coercivity. This is an important parameter to help quantify the nonlinearity of a hysteresis loop. We can see that this quantity is zero only in the case of coherent rotation with an applied field exactly

perpendicular to easy axis as illustrated in **Fig. 1.7** below. In the case of the linear sensor design, we apply the field perpendicular to the free layer easy axis (θ= 90°) as defined by shape anisotropy. In the large area sensor, on the other hand, we apply the field parallel to the free layer easy axis (θ= 0°) as defined by induced anisotropy.

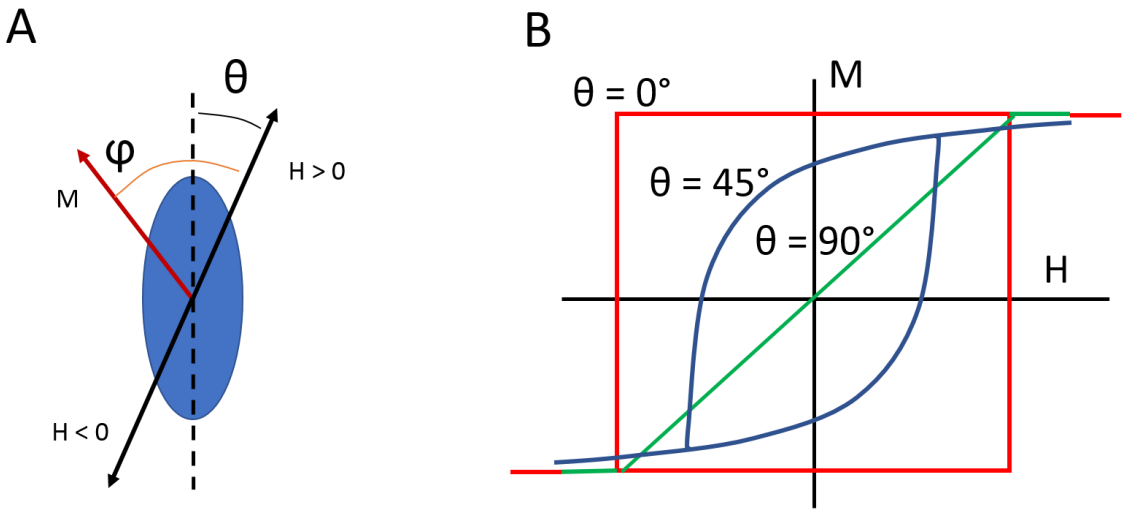

Fig. 1.7. Stoner-Wholfarth model of coherent rotation against a uniaxial easy axis. A) θ defines the angle between easy axis and applied field. φ defines the angle between magnetization and applied field. B) Some examples of hysteresis loops generated by the model for different values of theta.

4.3.2. Non-coherent switching modes

It is often impractical to work with objects that adhere to coherent rotation during magnetic switching. There exist many possible switching modes once we relax the assumption of single crystal ellipsoidal ferromagnetic objects. In such cases, we can hypothesize the switching behavior based on micromagnetic models. When such models are compared to experimental observations, we can begin to classify and characterize certain useful structures such as domain walls and nucleation sites. This becomes especially critical when considering switching in the free layer of the large area sensor, which is quite complex and believed to be dominated by domain nucleation. In such cases, it is best to use micromagnetic simulation software such as OOMMF [63].

4.4. Domain walls

4.4.1. Minimization of Zeeman energy

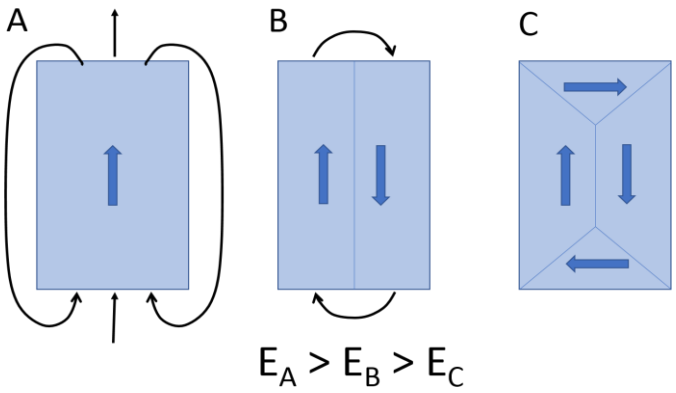

Fig. 1.8. Zeeman energy can be reduced at the expense of increased exchange and crystal anisotropy energies by breaking up configuration A into multiple domains such as B or C.

Under zero external applied magnetic field, the most energetically favorable configuration for magnetization in terms of internal Zeeman energy due to the demagnetizing field is a closed loop. Due to crystal anisotropy, a simple closed loop is usually not possible. A much more common structure is composed of either two or four magnetization directions that approximate a closed loop as illustrated in **Fig. 1.8**. Each of the coherent magnetization volumes within such a structure is known as a domain, and the boundary between two adjacent domains is known as a domain wall.

4.4.2. Crystal anisotropy vs. exchange energy

The domain wall has a finite width due to the exchange energy of adjacent atoms. As shown in **Fig. 1.9**, each atom within the domain wall is forced to turn slightly with respect to its neighbors. Materials with high exchange energy relative to crystal anisotropy will have wide domain walls. Conversely, materials with high crystal anisotropy relative to exchange energy will have narrow domain walls.

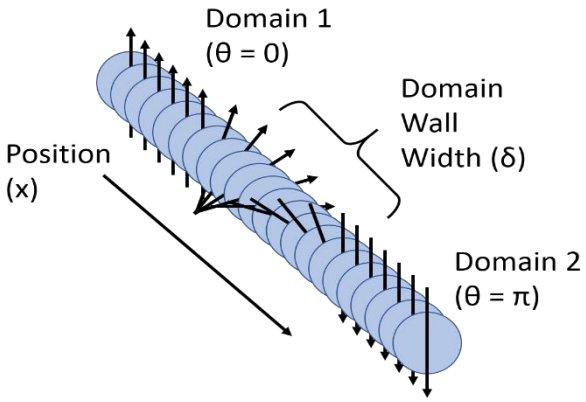

Fig. 1.9. Individual atoms in a domain wall are forced against crystal anisotropy by exchange coupling between neighboring atoms. Domain wall width relates to micromagnetic structure as defined by Eq. 1.2.

$$\theta(x) = \tan^{-1}\left[\sinh\frac{\pi \cdot x}{\delta}\right] + \frac{\pi}{2} \qquad \textbf{Eq. 1.2}$$

4.4.3. Domain walls in thin films

In bulk materials, the micromagnetic structure of domain walls is such that volume magnetic charge and resulting Zeeman energy can be zero. This can be understood from the volume integration of the magnetization divergence in a domain wall. In bulk, the domain wall is free to obtain a micromagnetic structure with zero divergence as illustrated

in **Fig. 1.9**. That is, when we move along any direction, we observe no change of that vector component in the magnetization. This condition is achieved by the Bloch Wall seen in **Fig. 1.10**.

As magnetic structures are reduced in thickness, they reach a point when the Bloch Wall creates enough surface magnetic charge to offset the energy savings achieved by lack of volume magnetic charge. At this point, the micromagnetic structure of the domain wall takes on a structure known as the Néel wall. A special case can occur that is a sort of hybrid between the Bloch and Néel wall structures, but this is not important for the present discussion.

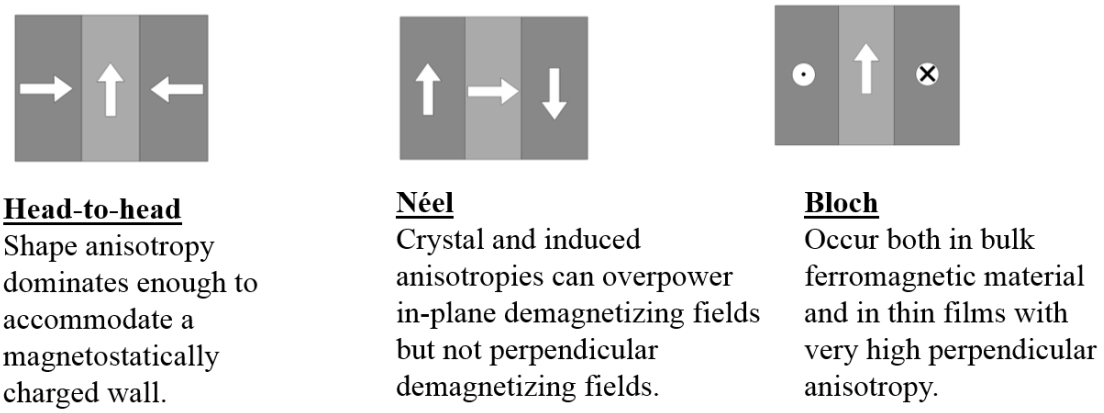

Fig. 1.10. Three possible domain wall structures.

4.5. Domain wall pinning

In certain cases, a geometrical defect, most commonly near edges, can harbor enough demagnetizing field energy to hold the local magnetization against exchange energy. In this case, the local variation in the demagnetizing field can be viewed as an energy minimum for domain walls. If the pinning energy minimum is strong enough, we can expect the pinned magnetization to remain even when the surrounding material is nominally saturated in the opposite direction. This can occur near rough edges of patterned films and likely reduces the magnetization uniformity in magnetic biosensors. In the

context of sensing MNPs, there is a weak pinning potential between the domain wall and MNP as discussed in chapter 5.

4.6. Domain nucleation

In contrast to domain wall pinning, new magnetic domains can be created spontaneously due to local energy irregularities. This process is known as domain nucleation [64]. Domain nucleation tends to be preferred in systems with high saturation magnetization and therefore high Zeeman energy. The original implementation of the large area multidomain switching GMR biosensor fits this criterion [19], but the newly deposited films were not behaving as expected. One goal of future work will be to understand how to reproducibly enhance domain nucleation in the large area multidomain switching sensor. The best form of analysis for domain nucleation is micromagnetic simulation [65, 63].

4.7. Superparamagnetism

As already noted, domain walls occur as a means of minimizing demagnetizing energy. They also have a finite width due to the required balance between crystal anisotropy and exchange energies. This domain wall width plays a significant role in the discussion of MNPs because, as the size of a single magnetic crystal is reduced, it will eventually cross the limit of domain wall width as defined by the physical properties of that material. For this reason, we note that the magnetostatic energy and therefore the coercivity of a MNP increases as particle size decreases. This is a manifestation of reduced degrees of freedom for magnetic switching.

At the same time, we must note that the total crystal anisotropy energy scales with physical volume. As volume is reduced, it will approach the thermal energy k_BT and begin to experience thermal fluctuations. Therefore, as we continue to decrease the MNP size

below the domain wall width, we reach a maximum coercivity. The coercivity then begins to reduce as thermal fluctuations begin to dominate.

We can visualize superparamagnetism as in **Fig 1.11**. Since there are no domains, each magnetic nanoparticle is always saturated. On the other hand, energy fluctuations due to temperature cause the magnetization direction to rapidly and randomly switch. Consequently, the outward appearance is zero net magnetic moment even though the particle is highly magnetic. As we increase the external magnetic field applied to the superparamagnetic nanoparticle, we begin to see a net magnetization in the same direction as the applied field even though thermal energy is still generating rapid switching. Only when a strong magnetic field is reached (commonly in the range of 10kOe) do we see a saturation of the full magnetic moment along the applied field direction. At points between zero moment and full saturation, the superparamagnetic nanoparticle follows a curve known as the Langevin Function (**Eq. 1.3**).

$$\left\langle \bar{m}_p \right\rangle = m_0 \left(\coth \frac{m_0 H_t}{k_B T} - \frac{k_B T}{m_0 H_t} \right) \hat{H}_t \qquad \textbf{Eq. 1.3}$$

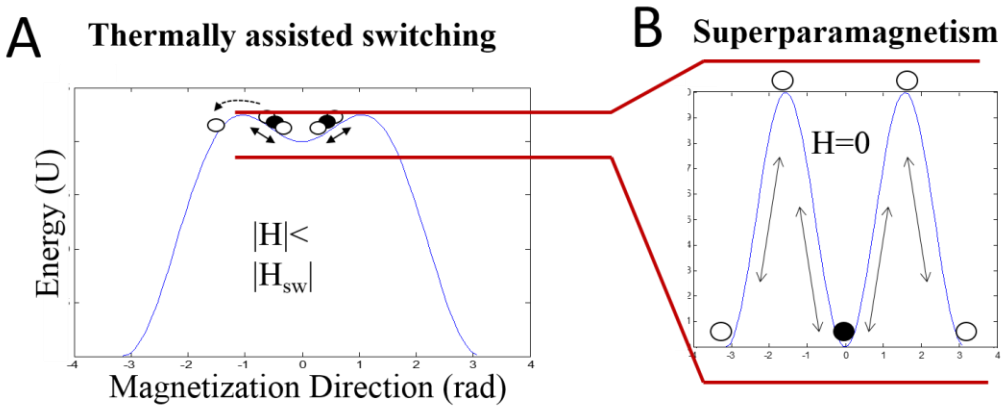

Fig. 1.11. A) In most cases, random energy fluctuations only serve to push the magnetization over a relatively small barrier to assist switching. B) If the physical volume of a ferromagnetic particle is sufficiently small, anisotropy energies are low enough to allow thermal energy to dominate. This case is called superparamagnetism.

If we take the first derivative of the Langevin function, we note that the slope of the curve through zero applied field is proportional to the saturation moment, m_0, of the superparamgnetic nanoparticle. This is a critical result for biosensing that leads to the technological importance of high magnetic moment superparamagnetic nanoparticles. In the low field regime where biosensing occurs (applied field +/- 30 Oe in this work), we note that the magnetic susceptibility of superparamagnetic nanoparticles is proportional to the square of saturation moment. This result was used quite effectively in previous work by our group with high-moment CoFe magnetic nanoparticles [66].

4.8 Magnetoresistance

4.8.1. Types of magnetoresistance

There are several types of magnetoresistance that arise from different mechanisms but ultimately derive from the spin of electrons affecting electrical conduction. This is used by engineers in magnetic field sensing in a variety of applications, solid state memory devices, and other important technologies. AMR is an effect that was first observed in 1856 by William Thompson. He discovered that the electrical resistance of a ferromagnetic material is at a maximum when the electrical current is along the same direction as the magnetization and is minimum when the current and magnetization direction are at right angles. The current is oriented in the plane of the film and so is called CIP geometry.

A very common application of AMR sensors is position sensing for automotive and industrial applications. The device is simply composed of μm-range thickness soft ferromagnetic films such as Permalloy ($Ni_{80}Fe_{20}$). The size of the AMR effect in transition metal ferromagnets is generally less than 5%, although recent research in new and exotic materials is showing promising results with much higher AMR values at room temperature [67]. In some cases, the noise of AMR sensors is low enough so that the signal-to-noise

ratio competes with that of GMR. It is possible with new developments in the future that AMR becomes the preferred sensing technology for biosensors.

TMR has become the preferred sensing technology for magnetic hard drive read heads [68]. This is because the signal-to-noise ratio is quite high with TMR approximately 200%. The TMR structure consists of fixed and free layers that are magnetic thin films (several nm-scale) and separated by a thin insulator such as Al_2O_3 or MgO. TMR is less desirable for biosensing applications because it requires electric current to pass vertically through a tunnel barrier and therefore requires electrodes on top and bottom. We call this geometry current-perpendicular-to-plane, or CPP. The top electrode must have sufficiently low resistance to avoid current nonuniformity issues, and so we cannot bind MNPs very near the sensing layer.

GMR is generally preferred for magnetic biosensing because it is practical to achieve GMR as high as 20% at room temperature, and the overall signal-to-noise ratio usually lies between AMR and TMR sensors. Both CPP and CIP structures are possible with GMR, but CIP is the preferred structure for biosensing so that the sensing layer may be as close as possible to MNPs. The relationship between MNP height and signal strength is complex but generally lies within the range of 1/d to $1/d^3$, where d is the height of MNPs. This will be addressed in more detail for different sensor configurations throughout the dissertation.

4.8.2. GMR working principle

The working principle of GMR is based on spin polarization of electrons combined with spin-dependent electron scattering at the surface of ferromagnetic thin films. The GMR structure consists of two or more thin film layers of ferromagnetic material such as CoFe separated by a metallic thin film with a long mean free path, such as Cu. The long mean free path ensures that the electrons originating from one ferromagnetic layer will

survive long enough to interact with another ferromagnetic layer on the other side of the normal metal. A complete description of GMR requires quantum mechanics, but a simplified description is attained by the Mott Two-Channel Model [61, 62].

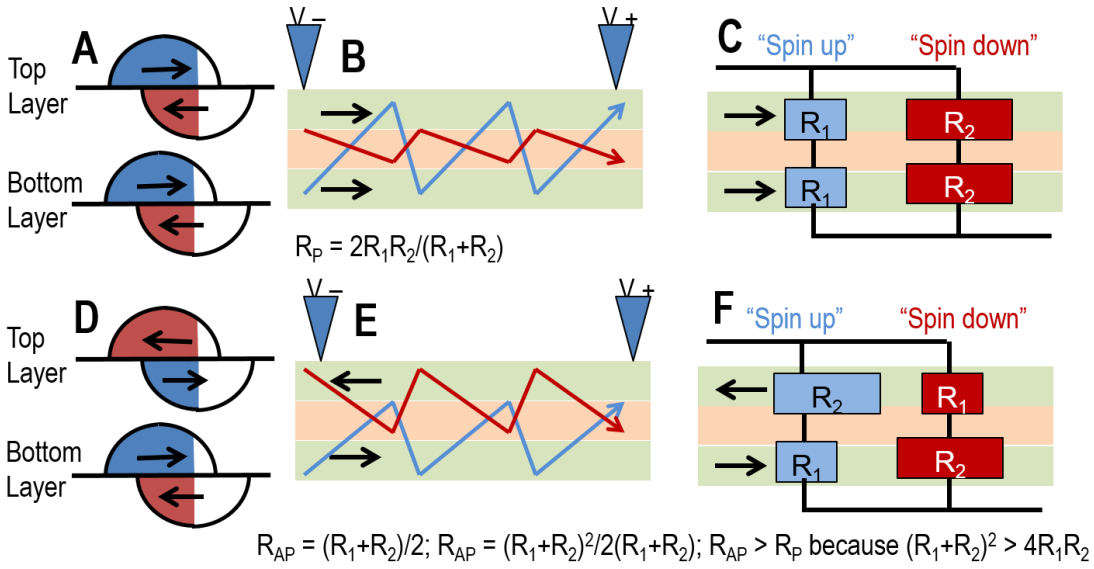

Fig. 1.12. Mott Two-Channel model. Spin states "up" and "down" refer globally to the fixed layer for convenience. A, D) Majority of states at Fermi level occur in the minority spin band of each ferromagnetic layer. B, E) Electrons with majority of states at Fermi level are more likely to scatter. C, F) Since scattering events are unlikely to flip spin, both spins compose two distinct channels of conduction.

The Mott Two-Channel model and a cartoon drawing of spin-dependent scattering are illustrated in **Fig. 1.12**. The conduction of spin-up and spin-down electrons can be considered as separate but parallel and therefore follow the same rules as any two parallel paths in electrical conductivity. This assumption is usually valid since the probability of spin-flip scattering is quite low compared to scattering events where spin is conserved.

Next, we observe that scattering is proportional to the DOS within ferromagnetic materials [62]. Since DOS at the Fermi Level is greater for spin-up electrons in our simple cartoon model, the conductance of the spin-down channel is less. By adding the individual conductance of both spin channels and converting to resistance, we end up with a very

familiar formula for combining any two resistors in parallel. This can be done for the case of both parallel and antiparallel configurations to compare their resistance. In the comparison, we conclude that $R_{AP} > R_P$ because $(R_1 + R_2)^2 > 4R_1R_2$.

4.9. Spin valve

The spin valve structure is a GMR device with a single free layer and fixed layer. The spin valve is known to have superior signal-to-noise ratio but lower performance in measuring precise vector components when compared to the antiferromagnetically coupled multilayer GMR [62]. Since the signal-to-noise ratio was our biggest concern in the biosensing application, we chose to work with the spin valve structure.

4.9.1. Fixed layer

Exchange energy is critical to spin valve biosensors as a means of fixing the direction of one layer during the fabrication step known as annealing. In this process, we raise the temperature of the spin valve above the blocking temperature of the antiferromagnetic layer. Consequently, the exchange energy causes the antiferromagnetic layer to orient itself along the direction of the adjacent ferromagnetic layer along the interface. A magnetic field can be used to define the direction of the ferromagnetic layer during this process. This field is held on during the cooling of the antiferromagnetic layer so that the interfacial exchange energy now holds the adjacent ferromagnetic layer fixed.

4.9.2. Dipole and orange peel coupling

Dipole and orange peel coupling both originate from fields on the surface of the fixed layer of GMR spin valves. In the case of dipole coupling (**Fig. 1.13A**), we note the stray field from north to south magnetic polarity in the fixed layer. This field biases the free layer in the antiparallel configuration and therefore shifts the transfer curve such that the zero-

field resistance is elevated. In addition, we note that the dipole field is strongest near the sensor edges, and so this adds nonuniformity to the free layer.

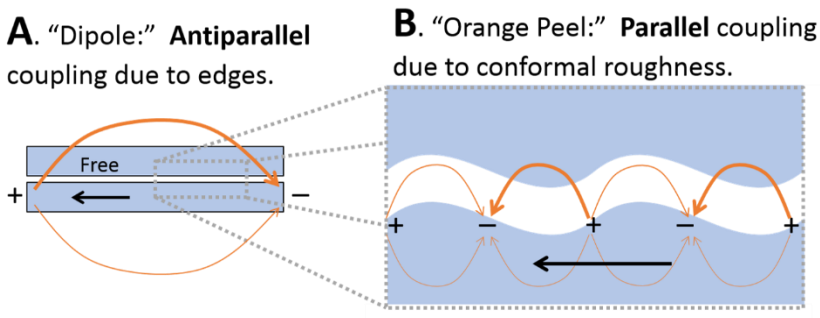

Fig. 1.13. A) Dipole coupling occurs due to the magnetically charged ends of the fixed layer. **B)** Orange peel coupling occurs due to magnetically charged surface roughness in the fixed layer.

Orange peel coupling originates from the microscopically rough or wavy fixed layer surface as illustrated in **Fig. 1.13B**. If the free layer has conformal roughness or waves, then we get a stray field coupling that tends to shift the free layer towards the parallel configuration. This coupling depends very strongly on the spacing between the free and fixed layers, so it can be tuned in GMR spin valves by adjusting the copper thickness. Orange peel coupling creates a shift of the transfer curve opposite that of the dipole shift.

This effect is generally quite uniform across the sensor surface, so its strength does not depend on sensor size. Since the dipole field decreases rapidly for wide sensors, the orange peel coupling is more dominant in this case. We tend to notice orange peel coupling in the large area sensor.

4.10. Detection of MNPs

The detection of MNPs in GMR biosensors is derived from a shift in transfer curve due to the stray field emanating from MNPs. In the simplest model, we can calculate the MNP moment and consequent dipole field from excitation in the applied field and then superimpose the MNP stray field on the sensor as illustrated in **Fig. 1.14** below. In the conventional model that works for linear sensors, we integrate this MNP stray field throughout the entire free layer volume. However, this method assumes uniformity of the free layer, so it does not work for the large area sensor. In addition, we have learned that the stray fields from the fixed and free layers of the GMR spin valve are significant and so need to be included in the total field acting on the MNP. This detail creates a situation where we must iterate through the calculation several times before converging on an answer. More details will be included in chapter 3.

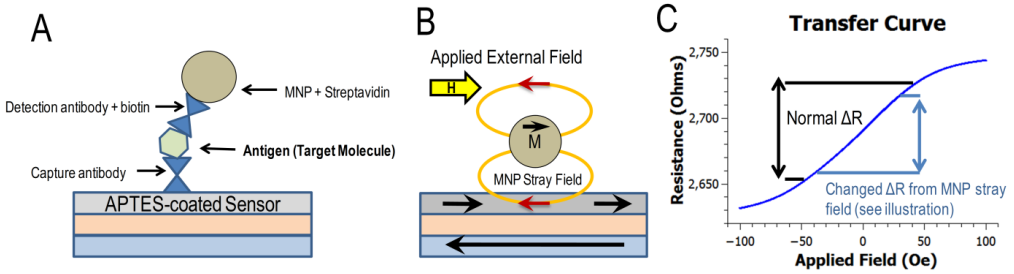

Fig. 1.14. A) Schematic of specific bonds provided by antibody / antigen pairs. The magnetic nanoparticle remains bound only if other bonds are already in place. B) MNP generates a magnetic stray field onto the GMR sensor. C) Signal derived from modified resistance versus applied magnetic field.

Chapter 2. Fabrication, Testing, and Surface Functionalization

1. Introduction

The fabrication and testing of GMR biosensors was carried out by the author in facilities at the University of Minnesota. This includes GMR spin valve film deposition and characterization as well as lithographic and thin film processing techniques for construction of GMR spin valve sensors. Surface chemistry and assay optimization was carried out by other graduate students and described in Appendix B.

2. GMR film deposition and characterization

The GMR films were deposited on a Shamrock sputtering system at the University of Minnesota in the Laboratory of Nanomagnetism and Quantum Spintronics. The Shamrock system includes five DC sputter guns, one RF sputter gun, an integrated ion mill, a planetary stage with DC magnetic field applied to each of the four substrate holders, a robotic sample loading system, and a heated load lock / cassette with space for twelve substrates. Several photographs of the Shamrock interior are included in **Fig. 2.1.**

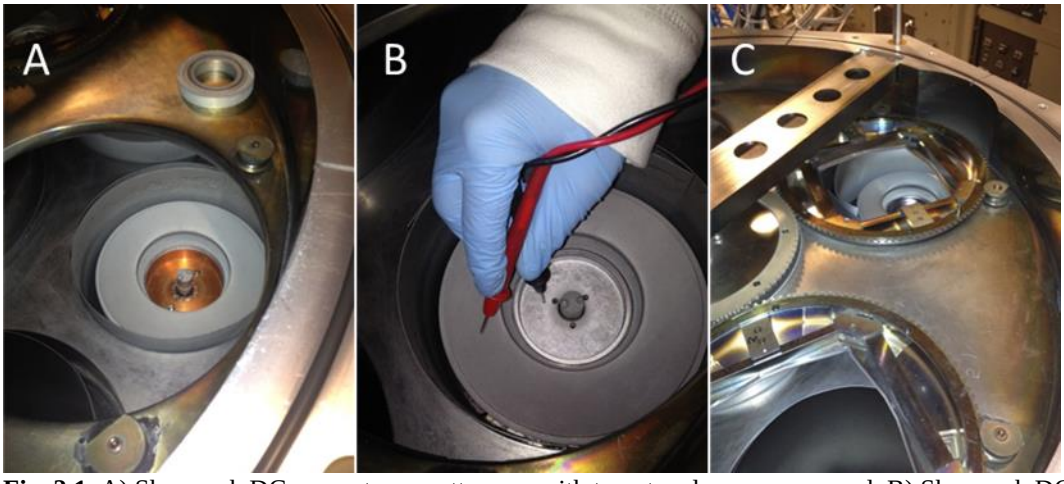

Fig. 2.1. A) Shamrock DC magnetron sputter gun with target and spacer removed. B) Shamrock DC magnetron sputter gun with target installed. C) Shamrock DC magnetron sputter gun visible beneath one of four substrate holders on the planetary stage.

Optimization of material deposition involved selection of materials as well as deposition conditions including pressure, power, and time for each material. Based on previous experience, we began with the following layer structure for the GMR film: Ta (30 Å) / IrMn (80 Å) / CoFe (30 Å) / Cu (33 Å) / CoFe (10 Å) / NiFe (20 Å) / Ta (50 Å). To maximize the overall sensor signal, we systematically changed deposition power and time for Cu and chose the recipe that produced the highest magnetoresistance on un-patterned film. We then deposited the materials on 20 wafers according to the established recipe for each material and the new recipe for Cu. The number of wafers was chosen in anticipation of the number of wafers needed plus some extras for optimization of the fabrication process.

To determine the quality of film deposition conditions, we employed two basic techniques:

1. VSM: This device uses a large electromagnet to apply a magnetic field. At the same time, the sample is vibrated between two small coils so that a voltage is induced according to Faraday's Law. This voltage ultimately depends on the magnetic moment of the sample.

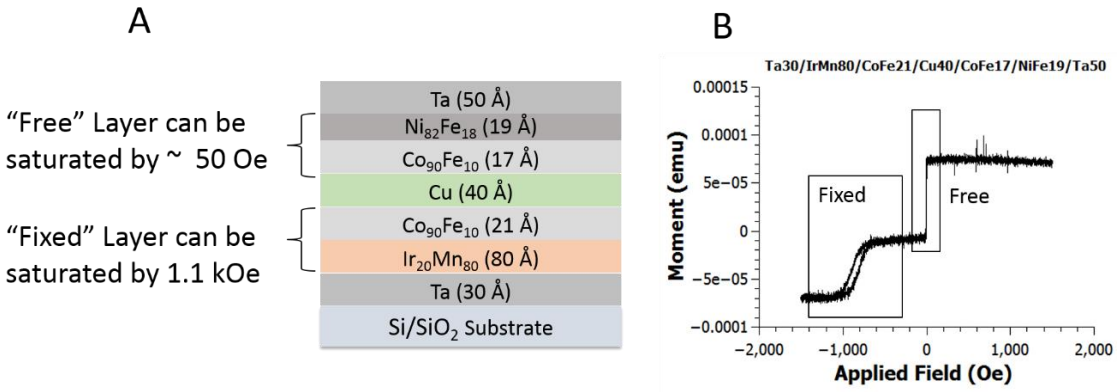

Fig. 2.2. A) Layer structure for GMR biosensors in this work. B) VSM data shows switching characteristics of free and fixed ferromagnetic layers in the un-patterned GMR biosensor film.

In the case of a good GMR spin valve, the moment of both the free and fixed magnetic layers can be clearly seen as in **Fig. 2.2B**.

2. Four-Point Probe: The GMR films can be tested electrically prior to biochip fabrication. The procedure is to apply a known electrical current, sweep the magnetic field with an electromagnet, and then read the voltage. The GMR ratio of the film can be obtained by dividing the voltage change by the minimum voltage during the sweep. A good example is shown in **Fig. 2.3A**. The comparison of several films is shown in **Fig. 2.3B**.

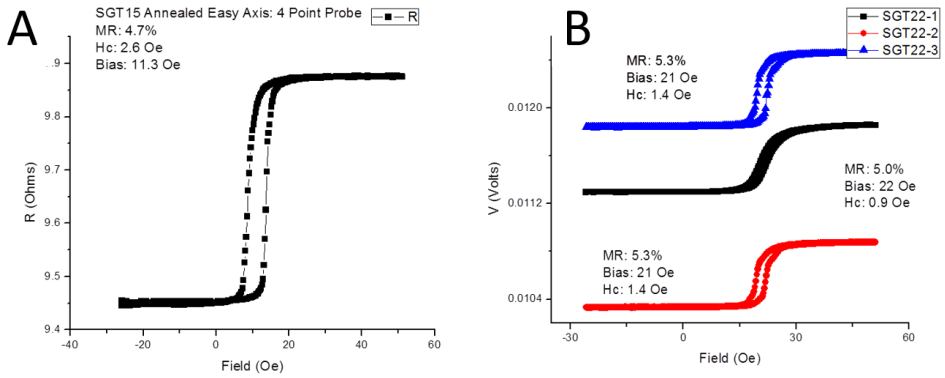

Fig. 2.3: Example of four-point probe measurements to confirm the electrical performance of two different GMR film samples: A) Wafer SGT-15 shows near easy-axis multidomain switching as preferred in the large area sensor (see chapter 4). B) Wafer SGT-22 shows more coherent switching and improved GMR ratio due to different deposition power of the copper spacer. The three plots in this case are taken from different locations on the same wafer.

3. Biochip fabrication and characterization

The first four steps of fabrication are illustrated in **Fig. 2.4**. The first step was lithography of the sensor shape. This was the step that defined whether we would fabricate a large area sensor, a domain wall sensor, or a linear sensor. In the case of a large area sensor or a linear sensor, this is a photolithography step performed on the Canon 2500 i3 Stepper. In the case of a domain wall sensor design, the features were too small for photolithography and required the use of the Vistec EBPG 5000+ electron beam pattern generator. In all cases, the goal was to define a temporary pattern with resist. In the next step, we used the ion mill

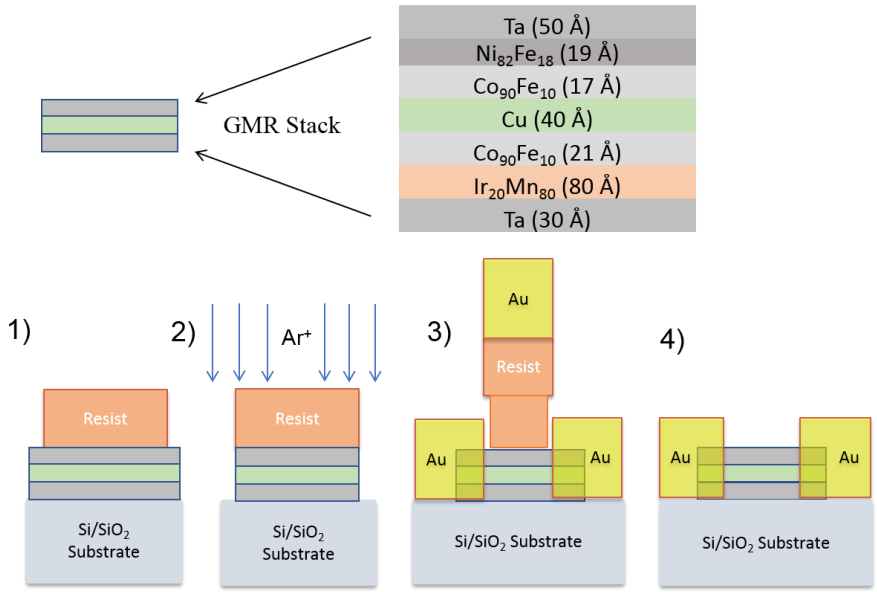

Fig. 2.4. Fabrication steps. 1) Photolithography, positive photoresist; 2) Ion milling, resist removal; 3) Photolithography, positive bi-layer resist, Cr / Au / Cr deposition with e-beam evaporator; 4) Liftoff deposited Au to create electrodes (250 Å Cr / 2.5 k Å Au / 500 Å Cr).

to physically etch all material not protected by resist. After the etching, we sometimes included an optional step (not shown in figure) of backfilling the etched area with evaporated SiO_2 from the Varian electron beam evaporator before removing the photoresist to expose the patterned GMR sensor. Next, we created electrical connections between GMR sensors and the edge of the biochip by patterning a bilayer resist (either photo or e-beam) and evaporating a multilayer Cr / Au / Cr with appropriate thickness for the application. In the case of the large area sensor and linear sensor, we found that Cr 50A / Au 2500 A / Cr 50A worked well. In the next step, we performed liftoff by dissolving the resist in solvent. Finally, we patterned thick SiO_2 over the top of electrodes and thin Al_2O_3 + SiO_2 over the sensor surface.

4. Magnetic nanoparticle characterization

MNPs were characterized with a VSM in the Institute of Rock Magnetism at the University of Minnesota. Two different MNP samples were used for comparison. The first was a 50nm MACS bead from Miltenyi Biotec [69] composed of approximately 10-15

MNPs held together by a polymer matrix. The second sample was from Ademtec and was a 200 nm bead composed of approximately 1000 MNPs held together by a polymer matrix. Results were analyzed by a Python script that normalized the measured magnetic moment from VSM data and fit the curve to a Langevin function.

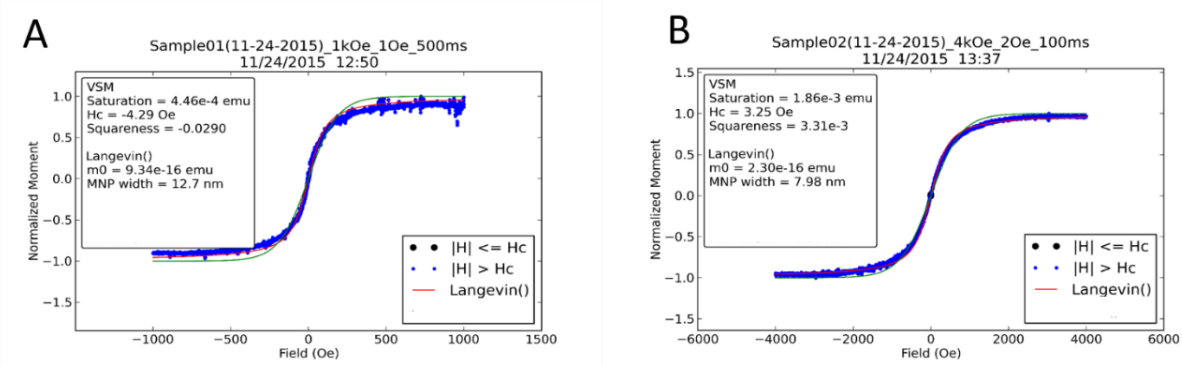

Fig. 2.5. A) Normalized VSM data fit to Langevin function indicates each MNP inside 50nm MACS beads is approximately 13nm if Fe_3O_4 is assumed. B) Normalized VSM data fit to Langevin function indicates each MNP inside 200nm Adembead beads is approximately 8nm if Fe_3O_4 is assumed.

From the Langevin Function fitting data in **Fig 2.5**, we can determine the moment of each individual MNP inside the beads and infer the size of MNPs by assuming that the MNP saturation magnetization is the same as that of bulk Fe_3O_4. We know this underestimates the size of MNPs because there is usually a thin magnetically dead layer on the surface of each MNP. This metric is quite useful for visualization purposes, however. We note that the moment of each MNP is extremely critical in the biosensing application because it gets squared in the low-field susceptibility. The magnetic field from the MNP is proportional to the saturation moment multiplied by the Langevin function, and the slope of the Langevin function near zero field is proportional to the MNP saturation moment. Since biosensing is indeed a low-field application, we get a very significant advantage from high-moment MNPs [66].

Chapter 3. Linear Sensor

1. Introduction

As stated above, our major contribution to this area of the GMR biosensing field was to discuss the design considerations of non-uniform signal strength due to the location of MNPs relative to the sensor edge. The MNPs close to the edge of each sensing strip contribute a large signal while the MNPs close to the center of each sensing strip contribute a small signal. This occurs for two reasons: First, we note that the dipole field of a MNP is symmetric along the direction of the total applied field, and the signal strength equals zero across an infinite plane. Only by breaking the symmetry do we see a signal. Second, the GMR free and fixed layers produce a strong stray field on the MNPs. This stray field exceeds the external applied field in cases where the MNP is very close to the sensor edge. For both reasons, we need to carefully consider the impact on sensor design.

1.1. Motivation for use of a linear sensor

Development of biological assays including the ovarian cancer multiplex assay on a GMR biosensing requires a multidisciplinary effort and multiple team members. For this reason, it was practical to base assay development on a robust sensor design which was already popular in the literature and had a quantitative model to predict performance. In this chapter, we will discuss how we optimized the performance of what is commonly called the linear GMR biosensor.

In the context of magnetic sensors literature, linearity refers to a sensor's response to an externally applied magnetic field. It is important to point out that the assumption of linearity is only an approximation and there are several nonlinear effects which need to be minimized to optimize biosensor performance. In this chapter, we will begin by reviewing

some basic design considerations, move on to an analytical model which is useful for understanding the special considerations related to magnetic biosensing, and finally present a micromagnetic simulation that enabled us to fine tune our understanding.

1.2. Linearity approximation

To approximate linearity in this sensor, we must begin with an understanding of magnetic switching in thin films as described in previous chapters before moving on to more subtle details. As described by the Stoner-Wohlfarth model [59], a uniformly magnetized ellipsoid with uniaxial anisotropy will exhibit a linear response to an applied magnetic field if the direction of the applied field is at a right angle to the easy axis. Any shape other than ellipsoid will exhibit more complex switching behavior due to a nonuniform demagnetizing field inside the body. Clearly this condition is not practical in a thin film device like a GMR sensor. To come as close as possible to the ellipsoid condition, we would need to match film thickness with width for a square cross section. In the case of a GMR sensor, the free layer thickness would have to be in the range of 3-5nm. This is far beyond the capability of lithographic patterning, where the minimum feature width is tens of nm for electron beam lithography or hundreds of nm for photolithography. Consequently, we know that coherent rotation is not possible and only a convenient approximation.

Additional deviation from sensor linearity comes from the effects of coercivity, the Oersted Field from sensing current, and orange-peel coupling. For a more complete description of each, please refer to these topics in Chapter 1. For now, we note that the observed coercivity in the GMR sensor is approximately 0.1 Oe. The Oersted Field is less than 1Oe at the sensor surface with the current densities used. Also, the micromagnetic

simulations found in the literature predict a slight parabolic component to the GMR sensor transfer curve due to orange peel coupling [70].

1.3. Biosensing specifications

Integration with the biosensing application puts some practical constraints on the linear GMR sensor design. For example, the common approach to maintain free layer uniformity for hard drive reader applications (and therefore common in the literature) is to construct permanent magnets on the ends of the GMR sensor [71]. This is a very effective design and is known to improve the signal-to-noise ratio. Unfortunately, a permanent magnet like this would include a relatively strong field divergence and therefore attract MNPs to the surface of the magnet. This would non-specifically pull MNPs out of solution and deplete the tags from the assay, making MNP-based biosensing impossible. The most practical way to set the free layer easy axis, then, is to use shape anisotropy. For this reason, we constructed very long and narrow sensor strips.

Another practical limitation for GMR biosensing is that the sensing area needs to be covered by capture antibodies. With the use of an array spotting system, we can reliably print capture antibody solution with less than 1nL volume. Even with this impressive capability, the printed droplet spreads on the sensing surface to a diameter of approximately 250 μm. We tried to maximize the sensing area within the droplet in order capture as many magnetic tags as possible and thereby reduce statistical variation. At the same time, we noted that the middle of the droplet generally has more uniform print density than the edges. For this reason, we chose the length of our sensor to be 100 μm. Due to the concomitant constraint of shape anisotropy, each sensing area consists of many long narrow GMR strips so that the total width from the edge of the first sensor to the last is approximately 100 μm as indicated in **Fig. 3.1** below.

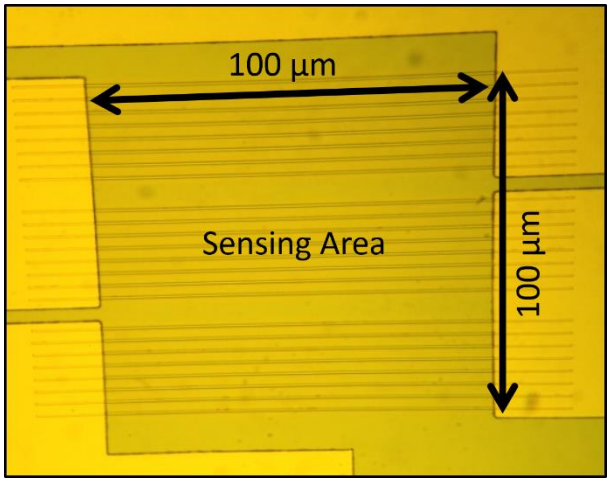

Fig. 3.1. The linear GMR biosensor is composed of several groups of strips to meet practical specifications.

1.4. Electrical specifications

Finally, we had to consider the practical design constraints imposed by the electrical readout of the GMR biosensor. We wanted to construct a sensor with reasonable resistance levels to keep the GMR as the primary voltage drop through the sensor when compared to the analog switch multiplexor (generally 5-10 ohms). We also wanted to keep the resistance relatively low to minimize Johnson noise. For these reasons, we chose a sensor resistance between 1kohm and 5kohm and therefore needed parallel connections across sensor element groups and series connections from one group to the next. This arrangement has the added benefit of reliability: if one GMR stripe is damaged during fabrication, the other parallel GMR stripes in each group add redundancy.

2. Theoretical model

Although we knew that the "linear" GMR sensor is not perfectly linear, we began with a linear model before moving to a more complex model. By assuming linearity and coherent rotation, we could derive analytical equations to guide sensor design. We tried to answer questions that impacted fabrication decisions as well as the role of MNP position and the stray fields from free and fixed layers. Our work was originally integrated into a MATLAB program assuming a uniform free layer but using real transfer curve data as input to account for the majority of nonlinearity. Later this program was rewritten in Python and included a micromagnetic structure calculation to add precision.

Various GMR sensor strips were modeled and judged by their ability to detect a uniform distribution of Fe_3O_4 MNPs. We found that the free and fixed layers of GMR sensors play a dominant role in exciting the MNPs, and consequent MNP stray fields lead to increased free layer susceptibility. This result persisted even when MNPs were confined to the interior of the sensing region. It was enhanced when MNPs were allowed near exterior edges. Our analysis included three different fabrication configurations on two different spin valve film stacks and agreed well with experimental results as shown in **Fig 3.6**.

The sensing of MNPs for biological applications is much different than other magnetic field-sensing applications due to the local variation of field strength across each sensing element. This leads to a position-dependence for the signal sign and strength related to MNPs fixed in a variety of locations across a sensor [56]. In addition, the sensing elements themselves generate stray magnetic fields that polarize nearby MNPs. This particular issue is exacerbated by magnetic label size reduction from 100's of nm down to 50nm or less and consequent proximity to the field-emitting regions near the edge of sensing elements [33]. Great care was taken in mathematical analysis to account for this issue, and the result was used to guide device design and fabrication.

Models for mean MNP stray field across the sensor have been proposed and compared to the experimental results [72, 37, 54, 73, 74]. The effects of magnetostatic fields from GMR free and fixed layers have also been discussed [75, 76, 77]. However, what was previously missing was a systematic description of how to incorporate these fields and use the information to guide fabrication and sensor optimization. As listed in **Table 3.1**, we analyzed three different fabrication configurations on two different spin valve films. For each of these, we included four different derivations (Boolean combinations of free and

fixed layer magnetostatic fields) to highlight the effect of stray fields from GMR sensing elements. We also determined the optimal DC offset field for the given AC field amplitude of +/- 30 Oe.

The schematic illustration of our simulated devices is shown in **Fig. 3.2B**. In **Fig. 3.2C**, we show just five MNP positions along the y-axis, but the actual calculation uses 100 positions. A uniform grid of positions was used with 40 nm spacing between each point.

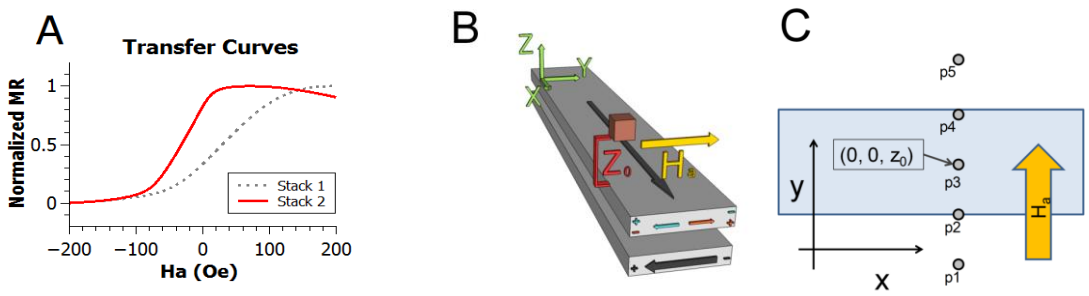

Fig. 3.2. A) Normalized GMR transfer curve from Film stack 1 and Film stack 2 used as input parameters for simulation. B) 3D schematic of GMR sensing element, representative MNP, and applied field. C) Top view schematic showing placement of several MNPs along y-axis (x=0). [55]

This uniform grid allowed us to focus attention only on MNP positions confined to the y-axis (x=0). This approach does not generalize to the case of non-uniform particle distributions, but it is acceptable for comparing different sensor designs. Each MNP's y-position generates a unique signal, but the dependence on x-position was assumed to be negligible due to the high aspect ratio of each device (length x = 100 µm, width y = 750 nm). Each MNP was 20 nm hydrodynamic diameter Fe_3O_4 as sold by Ocean Nanotech. From VSM data, each MNP was determined to be superparamagnetic with a saturation moment of 8.0×10^{-16} emu and an estimated magnetic diameter of 14.7 nm.

The height (z_0) of each MNP above the middle of the free layer was set by the MNP radius and any materials in between (e.g. 50 Å Ta, 300 Å SiO2 isolation, etc.). Three fabrication configurations are illustrated in **Fig. 3.3**. These differ only by the MNP height variation along the y-axis. In the case of fabrication configuration A, we assumed just 16.5

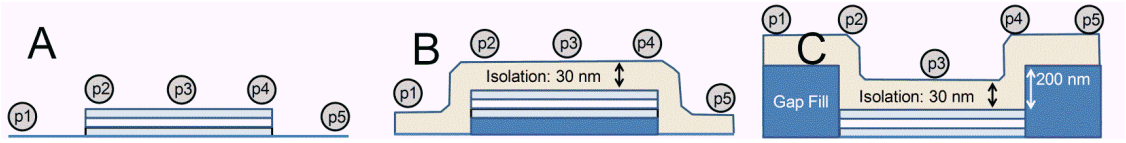

Fig. 3.3. A) Device configuration "A" represents the minimum spacing between MNPs and sensor (5 nm Ta plus MNP radius). B) Device configuration "B" inserts an isolation layer between MNPs and sensor. C) Device configuration "C" lifts by 200 nm all MNPs outside the GMR sensing region. [55]

nm spacing from the MNP dipole center to the center of the free layer for the region directly above the GMR sensing element. For other regions, the MNP was assumed to be at the same height as the center of the free layer. Configuration B included an additional 30 nm of height due to isolation in the region directly above the GMR sensing element, with the other regions left at the same level as the middle of the free layer. Finally, configuration C is the same as B but with additional height of 200 nm for the regions not directly above the sensor.

Table 3.1: Description of each combination of (Film stack: Configuration: Derivation) in the naming scheme. [55]

Derivation (Stray Fields)	Film stack	Config: Fig. 3.3A	Config: Fig. 3.3B	Config: Fig 3.3C
Fixed Layer: 0 (no)	**Fig. 3.2A**: Stack 1	**1A-00**	**1B-00**	**1C-00**
Free Layer: 0 (no)	**Fig. 3.2A**: Stack 2	**2A-00**	**2B-00**	**2C-00**
Fixed Layer: 0 (no)	**Fig. 3.2A**: Stack 1	**1A-01**	**1B-01**	**1C-01**
Free Layer: 1 (yes)	**Fig. 3.2A**: Stack 2	**2A-01**	**2B-01**	**2C-01**
Fixed Layer: 1 (yes)	**Fig. 3.2A**: Stack 1	**1A-10**	**1B-10**	**1C-10**
Free Layer: 0 (no)	**Fig. 3.2A**: Stack 2	**2A-10**	**2B-10**	**2C-10**
Fixed Layer: 1 (yes)	**Fig. 3.2A**: Stack 1	**1A-11**	**1B-11**	**1C-11**
Free Layer: 1 (yes)	**Fig. 3.2A**: Stack 2	**2A-11**	**2B-11**	**2C-11**

The first step of the simulation was the calculation of total field observed by each MNP. This total field included the stray fields from both the fixed and free layers as these were

found to be significant. For the calculation of GMR layer stray fields, we assumed infinite extent along the x-axis and magnetic charge confined to the x-z sensor surfaces as illustrated in **Fig. 3.4**. This leads to **Eq. 3.1** [60]. To derive the pre-factor σ, we needed to assume a magnetization Ms and then calculate its component along the y-axis. For the fixed layer, Ms = 1500 emu/cm^3. For the free layer, M_s = 1000 emu/cm^3 due to the strong coupling between $Co_{90}Fe_{10}$ (1500 emu/cm^3) and $Ni_{82}Fe_{18}$ (800 emu/cm^3).

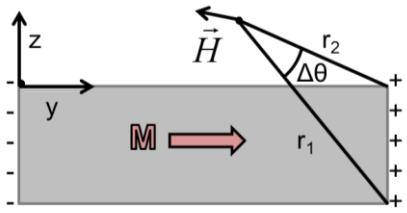

Fig. 3.4. Schematic illustration of **Eq. 3.1** for one edge of the film. [55]

$$H_y = -2\left(\vec{M}\cdot\boldsymbol{n}\right)\Delta\theta = -2\sigma\Delta\theta \qquad \textbf{Eq. 3.1}$$

$$H_z = -2\left(\vec{M}\cdot\boldsymbol{n}\right)ln\frac{r_2}{r_1} = -2\sigma ln\frac{r_2}{r_1}$$

For the n-axis (parallel or antiparallel with y) component of free layer magnetization, we noted the sin φ dependence of GMR where φ is the relative angle between free and fixed layers. In addition, we assumed a monotonic variation of the free layer angle versus applied field. For film stack 2 (**Fig. 3.2A**), the slight decrease in magnetoresistance after saturation represents the imperfect pinning of the fixed layer. To approximate the real situation, we took the simple approach of free layer saturation beyond the experimental maximum MR value. After that, we allowed the fixed layer to rotate to account for the calculated relative angle φ. This issue did not occur in film stack 1, so the film stack 1 results were expected to be slightly more accurate than those for film stack 2. The GMR layer stray fields were found to be similar in magnitude to the applied field itself and were therefore quite important to the derivation. In contrast, the MNP-MNP interactions were

expected to be small for cases of interest. In addition, the Oersted field induced by current through the sensor strips was ignored because the experimental design uses an AC current whose frequency does not match that of the applied AC field. Furthermore, the amplitude of this Oersted field was estimated to be less than +/-1.0 Oe at the sensor surface and still less at the position of the MNPs.

The second step of the simulation was the calculation of MNP moment. As already noted, we could assume superparamagnetic behavior with $m_0 = 8.0\times10^{-16}$ emu. The magnetization direction would therefore remain directed along the local total field (derivation 00, 01, 10, 11), and the moment would obey the Langevin Function (**Eq. 3.2**). All experiments and simulations were performed at T = 293 K. The symbol H was replaced by H_{00}, H_{01}, H_{10}, or H_{11} as appropriate for the given derivation.

$$m = m_0\left(coth(s) - \frac{1}{s}\right); s = \frac{m_0 H}{k_B T} \qquad \textbf{Eq. 3.2}$$

Next, we calculated the mean stray field from each MNP. Even though the distance was quite small between sensor and particle, we noted that the field was well approximated by a point dipole at (0, y_0, z_0) (**Fig. 3.2C**) because it is identical to that of a uniformly magnetized sphere. The mean field along the applied field direction was then simply proportional to the y-component of MNP stray field integrated throughout the free layer volume (V_f).

$$\bar{H}_y = \frac{1}{V_f}\iiint dV_f\left[\left(\frac{3(\vec{m}\cdot\hat{r})\hat{r}-\vec{m}}{r^3}\right)\cdot\hat{y}\right]; r = \sqrt{x^2 + (y-y_0)^2 + (z-z_0)^2} \qquad \textbf{Eq. 3.3}$$

The result of definite integration across a free layer of thickness t, length w_x, and width w_y is adapted from previous literature [33]:

$$\bar{H}_y = \frac{m_x}{w_x w_y}\left(\frac{1}{\sqrt{{X_-}^2+{Y_-}^2+{z_0}^2}}+\frac{-1}{\sqrt{{X_+}^2+{Y_-}^2+{z_0}^2}}+\frac{-1}{\sqrt{{X_-}^2+{Y_+}^2+{z_0}^2}}+\frac{1}{\sqrt{{X_+}^2+{Y_+}^2+{z_0}^2}}\right)+$$

$$\frac{m_y}{w_x w_y}\left(\frac{-Y_-}{({Y_-}^2+{z_0}^2)}\left[\frac{X_-}{\sqrt{{X_-}^2+{Y_-}^2+{z_0}^2}}+\frac{X_+}{\sqrt{{X_+}^2+{Y_-}^2+{z_0}^2}}\right]+\frac{-Y_+}{({Y_+}^2+{z_0}^2)}\left[\frac{X_-}{\sqrt{{X_-}^2+{Y_+}^2+{z_0}^2}}+\right.\right.$$

$$\left.\left.\frac{X_+}{\sqrt{{X_+}^2+{Y_+}^2+{z_0}^2}}\right]\right)+\frac{m_z}{w_x w_y}\left(\frac{z_0}{({Y_-}^2+{z_0}^2)}\left[\frac{X_-}{\sqrt{{X_-}^2+{Y_-}^2+{z_0}^2}}+\frac{X_+}{\sqrt{{X_+}^2+{Y_-}^2+{z_0}^2}}\right]+\right.$$

$$\left.\frac{-z_0}{({Y_+}^2+{z_0}^2)}\left[\frac{X_+}{\sqrt{{X_+}^2+{Y_+}^2+{z_0}^2}}+\frac{X_-}{\sqrt{{X_-}^2+{Y_+}^2+{z_0}^2}}\right]\right)$$

Eq. 3.4

where m_x, m_y and m_z are MNP magnetization components; x_0, y_0, and z_0 are MNP position components; w_x and w_y are sensor strip widths along x and y axes (**Fig. 3.2C**); and $X_- = \frac{w_x}{2} - x_0$; $X_+ = \frac{w_x}{2} + x_0$; $Y_- = \frac{w_y}{2} - y_0$; $X_+ = \frac{w_x}{2} + x_0$.

For any given simulation with well-defined z_0 (**Fig. 3.2B**), this result depends on both y-position of particle (y_0) and applied field H_a. A sample result is illustrated in **Fig. 3.5B**. Since we had no control over the precise location of any one MNP on the sensor surface, we estimated the "field per MNP" by averaging the result along the y-position of each MNP. This yielded a value that can be plotted against the applied field Ha. **Fig 3.5** illustrates specific results for each of the configurations listed in **Table 3.1**.

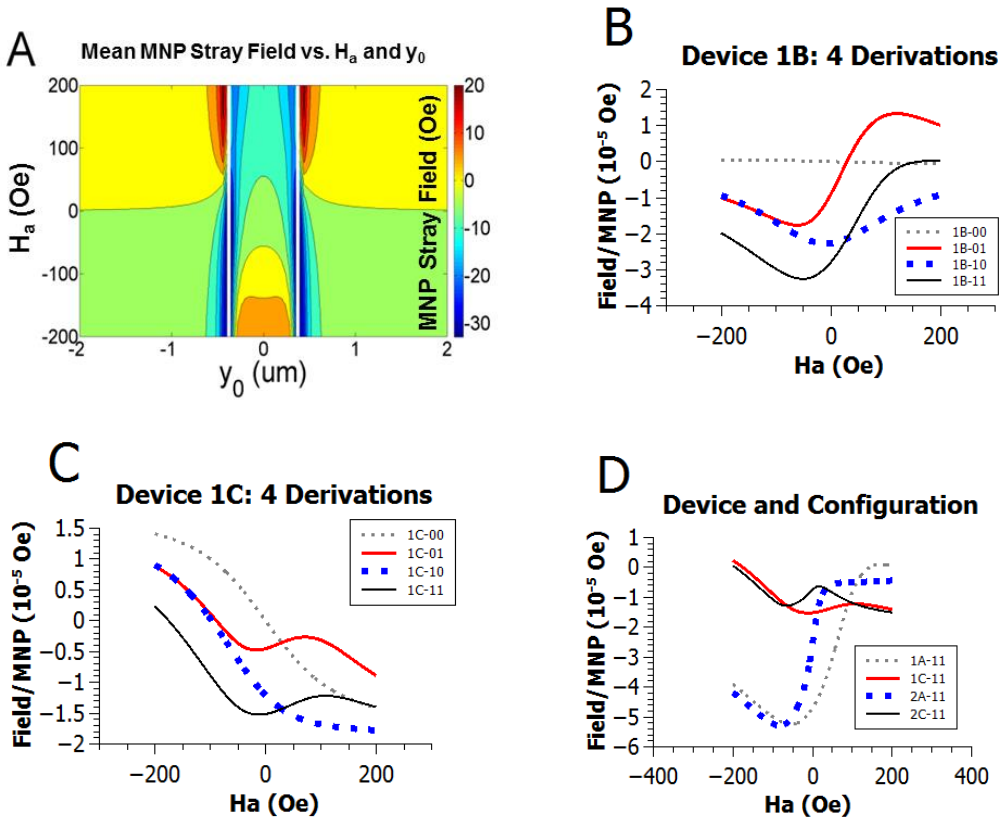

Fig. 3.5. A) Mean stray field across GMR sensing element plotted against different values of MNP position (y_0) and applied field (H_a). B) Mean stray field values from different derivations of device 1B averaged along y_0 plotted against applied field (H_a). This represents the stray field induced by an "average" MNP due to randomized location. C) Same calculation as in B but for different derivations of device 1C. D) Same calculation as in B but for different devices and configurations. Refer to **Table 3.1** for list of configurations. [55]

Finally, the signal was derived by multiplying the field per MNP by the total number of MNPs and then using the result to generate a resistance value from the original transfer curve. Here we simply interpolated by shifting each point of the transfer curve by the total MNP field at each value of H_a. For the result shown in **Fig. 3.6A**, we assumed 280,000 MNPs distributed throughout the entire 100 µm by 4 µm area. Here, 4 µm represents the pitch (along the y-axis) of a periodic array of identical sensing elements. Since a MNP stray field acts on the sensor free layer and the sensor free layer acts back on the MNP, it is necessary to iterate the calculation steps to converge on an accurate result of the

interaction between the sensor and MNPs. In our analysis, we found rapid convergence from 2.5% maximum error to 0.03% maximum error in just five iterations.

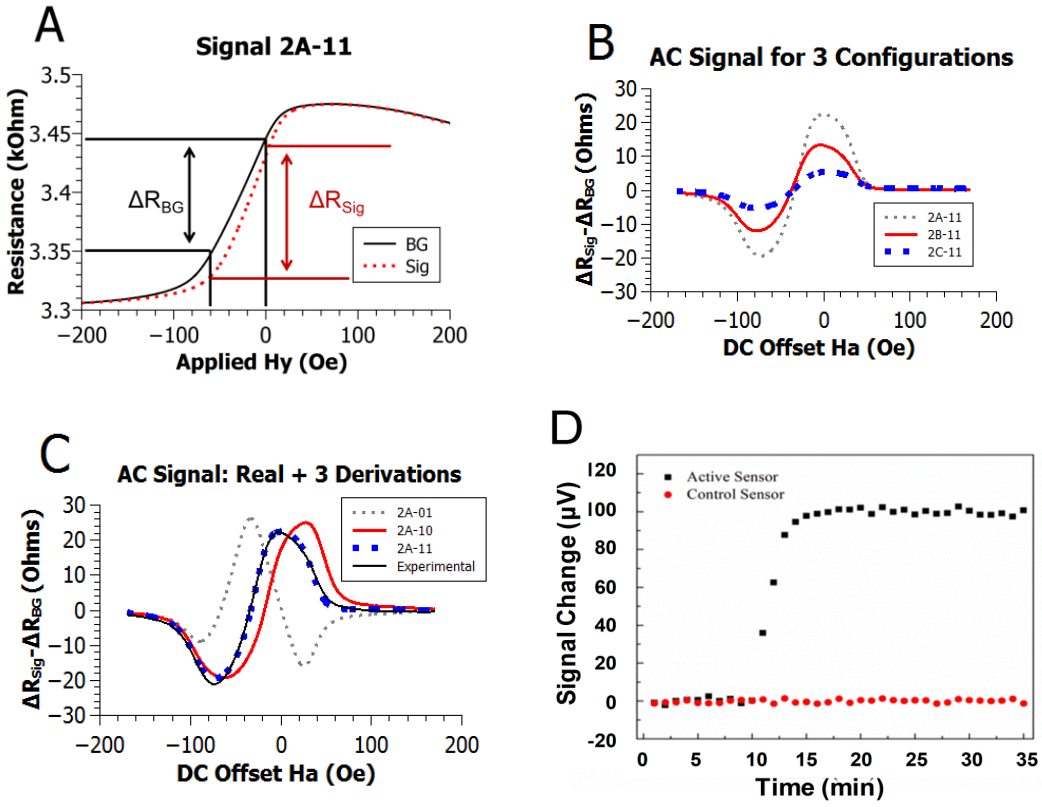

Fig. 3.6. A) Definition of ΔR_{BG} and ΔR_{Sig}. B) ($\Delta R_{Sig} - \Delta R_{BG}$) calculated versus DC offset field illustrating the role of GMR free and fixed layer magnetostatic fields. C) ($\Delta R_{Sig} - \Delta R_{BG}$) calculated versus DC offset field illustrating the role of fabrication configuration. D) Real-time detection signals. Black dots were signal changes from the active sensor and red dots were from the control sensor. [55]

In summary, the calculation steps were as follows: 1) Use the initial transfer curve (no MNPs) to estimate the free layer rotation angle and stray field. 2) Include applied field, free layer, and pinned layer stray fields in the total field seen by each MNP. 3) Use the field from MNPs to interpolate a new transfer curve from the initial one (no MNPs). 4) Use the new transfer curve to estimate the free layer rotation angle and stray field. Recalculate total field for MNPs. 5) Repeat steps 3 and 4 an arbitrary number of times and look for convergence.

In many cases, we may wish to apply an AC magnetic field to the sensor to obtain a fast measurement with low noise. For such a case, we can select the magnitude of AC field as well as the DC offset. In **Fig. 3.6A**, the signal from such an experiment is related to the change in ΔR values ($\Delta R_{Signal} - \Delta R_{Background}$). The example shown is for a +/- 30 Oe AC field centered on a DC offset of -30 Oe. **Fig. 3.6C** shows specific results for film stack 2 in various fabrication configurations and derivations. These results are plotted against DC field offset and assume the +/- 30 Oe AC field.

The simulation culminated in a plot of stray field per MNP versus H_a. This data can be further refined for direct comparison with experiment (**Fig. 3.6**), but also provides insight on its own (**Fig 3.5B-D)**. Deviations from derivation "00" indicate the importance of free and fixed layer magnetostatic fields. This deviation is most dramatic for fabrication configurations A and B but persists even when MNPs are confined to interior sensing regions as in configuration C. A positive slope represents local increase of free layer susceptibility. The comparison of different film stacks (**Fig. 3.5D**) illustrates the dramatic change induced by initial transfer curve. Here we note that the field per MNP is magnified by the film stack with larger initial free layer susceptibility (film stack 2). When this data is later used for transfer curve interpolation (**Fig. 3.6A**), higher initial susceptibility further magnifies the result. In terms of design considerations, the status of increased susceptibility is therefore elevated above the previous assumption.

3. Experimental results

As described in chapter 2, the GMR spin valve films were deposited with a Shamrock Magnetron Sputter System onto Si / SiO_2(1000) substrate: Ta(30) / IrMn(80) / $Co_{90}Fe_{10}$(25) / Cu(25) / $Co_{90}Fe_{10}$(10) / $Ni_{82}Fe_{18}$(20) / Ta(50), with thicknesses in parentheses in Angstroms. Spin valve strips were then fabricated by photolithography and ion milling of

the GMR film followed by a photolithography and liftoff of Cu contacts. Fabricated strip widths of 750 nm were confirmed by a JOEL 6700 SEM (**Fig. 3.7**). Each device was composed of several groups of parallel strips with 1.75μm pitch. Fabricated samples were annealed at 200°C for 60 minutes and then allowed to cool back to 30°C before removing the annealing magnetic field (H_{anneal} = -4 kOe).

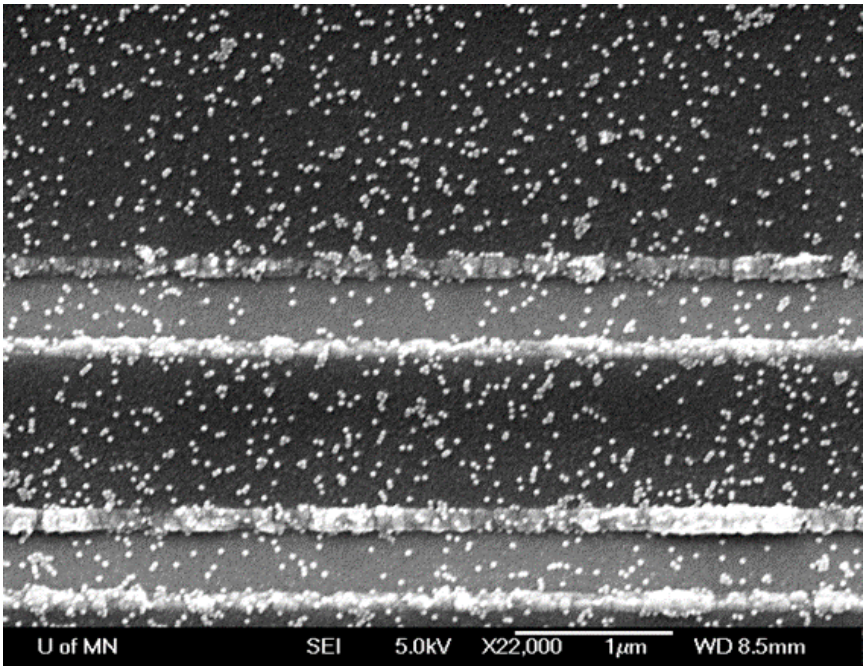

Fig. 3.7. Scanning electron microscopic image of PEG-coated 20 nm Fe_3O_4 MNPs from Ocean NanoTech bound to sensor surface.

Transfer curves for both film stacks are shown in **Fig. 3.2A**. The simulated performance of each film stack simply took the experimental curve (clean sensing surface, no MNPs) as an input parameter. During fabrication, the GMR sensor was coated with 10 nm Al_2O_3 and 20 nm SiO_2 layer by ALD, like configuration B (**Fig. 3.3B**). Experimental results in **Fig. 3.6A** and **Fig. 3.6D** used PEG-coated 20 nm Fe_3O_4 MNPs from Ocean NanoTech.

A typical experimental result is illustrated in **Fig. 3.6A**. At first, one might imagine that the positive change in susceptibility implies a high concentration of MNPs at sensor edge relative to center, but careful analysis proves this need not be true. This important fact may

be used to guide design, fabrication, and application of these sensors. We may allow MNPs to touch the sensor edge without worry of partial signal cancellation. In the case of designs used for this analysis, the MNPs on the sensor top and sensor edge all induce increased free layer susceptibility and therefore do not interfere with each other.

To apply the analytical calculation to sensor design and performance, we looked at two practical areas of interest: first, the effect of DC offset field used for AC H_a testing; and second, the effect of sensor pitch on sensitivity.

The analysis of DC offset was performed by calculating the change in ΔR values ($\Delta R_{Signal} - \Delta R_{Background}$) from transfer curves in **Fig. 3.6A**. Here, we held ΔH_a at 60 Oe (+/- 30 Oe AC field) and allowed the DC offset to vary. From **Fig. 3.6C**, we note that the theoretical result matched well with our experimental result. It is closer to derivation 11 than either 01 or 10, which indicates that both free and fixed layer stray fields play a significant role. Also, it is important to note that a specific experiment may yield positive, negative, or zero ΔR change depending on DC field bias from both the applied field and the fixed layer.

4. Discussion

Comparison of experimental results with theoretical analysis and simulation led to the conclusion that MNP excitation could be largely dominated by GMR layer magnetostatic fields. The same MNP positions that see high GMR stray fields (sensor top or edge near device boundaries) are also in the optimal position for signal generation. This led to several conclusions about design, fabrication, and application of GMR biosensors: First, we noted the elevated importance of free layer susceptibility. Conceptually, this parameter is squared in the calculation of $\Delta R_{Signal} - \Delta R_{Background}$. Second, we noted that MNPs along sensor edges need not be blocked because their signal interferes constructively with MNPs on the sensor

top. Our analysis shows that the sensitivity of configuration A (**Fig. 3.3A**) is best. On the other hand, this design is impractical due to a lack of surface passivation which is important for testing wet samples. Consequently, configuration B (**Fig. 3.3B**) is ultimately preferred. Finally, we noted the prediction of optimal DC offset for each sensor transfer curve depends on free and fixed layer magnetostatic fields but not fabrication configuration.

4.1. Linear sensor advantages

The main advantage of the linear sensor for GMR biosensor applications is the quantitative theoretical model that is already well-established in the literature. The design can therefore be modified and optimized with predictable results. This is an important criterion to consider, especially in the context of a multidisciplinary engineering challenge posed by biosensing systems. Another advantage of the linear sensor design is the simplicity of signal generation due to minimal hysteresis. Each magnetic field value approximately maps to a single resistance value. Even though GMR sensor resistance and magnetoresistance are known to drift with temperature, a simple approach to correction can be implemented by using a reference sensor as a baseline.

4.2. Linear sensor limitations

The linear sensor design does not naturally cover the desired amount of space in the biosensor as defined by the capture antibody printing spots. A large shape anisotropy is required to maintain near-linear performance, so many sensor strips must be grouped together with space in between. In addition, each MNP adds a different amount of signal depending on its location relative to the sensor strip edge. Likewise, binding of biological molecules is random and not correlated to the sensor strip features, so some variation is expected. Packing many sensor strips into the group to average out the MNP-location

variation works well when detecting a relatively large number of MNPs, but signal variation is expected to increase as the limit of detection is approached.

Another limitation to the linear sensor is simply the required trade-off between field-sensitivity and linearity. In order to minimize hysteresis, maintaining large shape anisotropy and a 90-degree field orientation as depicted in **Fig. 3.2B** (Stoner-Wohlfarth switching) is necessary. This limitation is unavoidable because it is intrinsic to the design.

4.3. Future directions for linear sensors

One way to combat the signal variation due to MNP location is to pattern capture antibodies so that they align more uniformly with the sensor strips. This way, specific binding on the assay would occur only in the predefined sensor locations to maximize signal and reduce variation. This would have a tremendous impact on the linear sensor performance, but the requirements for alignment in the sub-μm range are not trivial.

It is also possible that the sensitivity of linear sensors will be even further improved with the use of high-moment CoFe MNPs. It is important, then, to develop more reliable ways to consistently produce these nanoparticles in large quantities.

Chapter 4. Large area sensor with multidomain switching

1. Introduction

One important improvement that can be made in the context of biosensors for screening of ovarian cancer is increased sensitivity. The potential impact would be to help quantify new biomarkers at concentration levels that are at the edge of limit of detection for traditional technologies. In addition, greater sensitivity may reveal patterns in the biomarker data that were previously not recognized and possibly lead to the discovery of new biomarkers.

Due to previous success with the large area sensor with multidomain switching using high-moment CoFe MNPs [19], we intended to use a large area sensor with multidomain switching for our ovarian cancer project. However, the transfer curve of the sensor indicated magnetic switching dynamics much closer to coherent rotation than we had seen before. Both the multidomain and nearly coherent transfer curves are shown side by side in **Fig. 4.1**. Based on conventional models for the linear biosensor, we know that coherent rotation on a large area biosensor gives essentially zero response to MNPs, and, as expected from theory, we did not see any signal from biosensing experiments.

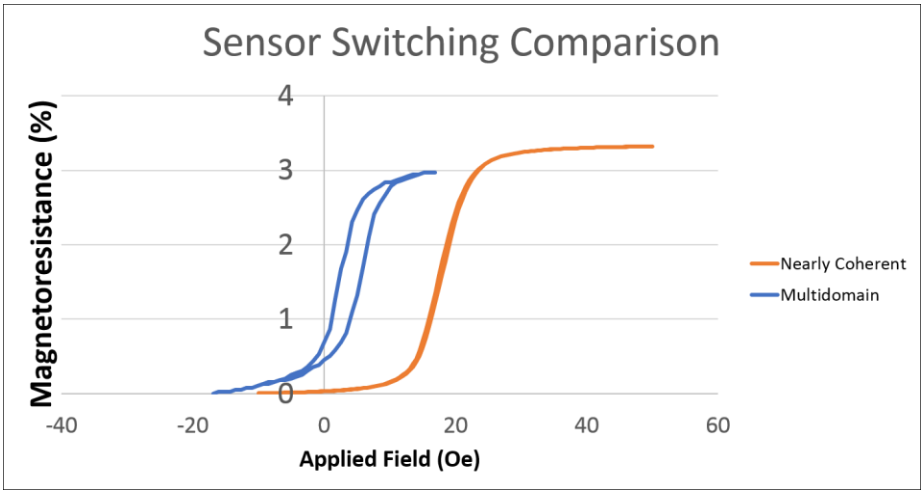

Fig. 4.1. Comparison between multidomain and nearly coherent switching observed in original large area and low aspect ratio sensor. The reasons for nearly coherent switching are unknown but must be related to GMR film deposition conditions.

Since our main goal was to develop the ovarian cancer assay, we elected to adopt the linear sensor design. At the same time, we worked to develop a theoretical model for a large area sensor to build on previous work and understand how to make our coherent rotation film behave more like the multi-domain film from previous work. This chapter will focus on our investigations related to this challenge. All analysis relates to the desired multidomain switching response because of the excellent experimental results.

2. Theoretical model

Signal levels in a magnetic biosensor can be enhanced by increasing sensitivity to magnetic field. One way to achieve this increased field sensitivity is by orienting the sensor so that an applied magnetic field is along the sensor easy axis. From the Stoner-Wohlfarth model, a coherent rotation in this case would yield a square hysteresis loop with infinite field sensitivity. Of course, a GMR sensor cannot really switch with coherent rotation because it does not adhere to the assumptions of the Stoner-Wohlfarth derivation. In this work, we allowed for complex switching behavior.

2.1. Initial approach and observations

In our large area sensor design, we began with a 40μm by 80μm rectangular area whose shape anisotropy was negligible because the demagnetizing field was very weak throughout most of the sensor. This was evidenced by the fact that orange peel coupling dominated the hysteresis loop offset. Instead of depending on shape anisotropy, the easy axis was determined by induced anisotropy from an applied magnetic field during GMR film deposition. Also, due to the large surface area of the sensor, we could safely assume a multi-domain configuration.

Three observations from our work with the large area sensor required explanation. First, we noted that the limit of detection for the large area sensor in terms of MNPs per square-micrometer was better than anything published about the linear sensor. At the time of publication, we attributed this performance to the use of high-moment CoFe particles plus the near easy-axis field orientation, but we still did not have a full theoretical understanding. Second, we noted that the conventional model used for analysis of the linear sensor predicted low sensitivity for the large area sensor. In that model, the assumption of uniform free layer with coherent rotation is required, so it clearly did not apply in the case of the large area sensor with multidomain switching. Finally, we noted that the sensitivity depended strongly (nearly $1/d^3$) on the height of MNPs. Again, this could not be accounted for by the conventional model in the literature because it was based on the condition of a uniform free layer. All three of these observations can be explained by the assumption that each MNP interacted only with a small and localized sensing area, but we did not have a theoretical model to explain this assertion.

Our first plan was to perform a full micromagnetic simulation with every detail, but this proved to be a challenge due to the number of micromagnetic elements required for a full simulation. When we approached another group that was well known for simulations,

they told us that the project was too large to manage. At this point, we decided to look for analytical methods related to multidomain switching. We first focused on domain wall motion, but the most up-to-date understanding was that the sensor switching was driven by a domain nucleation process [63]. The remaining portion of this chapter reports a simulation and experiment to enhance multidomain switching.

2.2. Sensor redesign

The multidomain switching array is related to previous literature in that we wished to enhance domain nucleation in a Permalloy thin film [78]. The main difference is that we chose to pin and de-pin domain walls for pseudo-nucleation switching dynamics. The precise structure represents a continuation of previous work with head-to-head domain walls (see chapter 5) [34,35]. We proposed, designed, and tested a multidomain switching array biosensor to address several challenges in the application of a large area sensor.

Three main specifications for the sensor were as follows: First, the sensor must cover a large surface area to match the capability of fluid dispensing systems for chemical activation of the sensor surface to immobilize the MNPs. Next, the sensitivity to field must meet or exceed the sensitivity from previous work with the large area sensor [19]. Finally, the sensor must have a quantifiable theoretical model. The multidomain switching array was micromagnetically modeled and fabricated as a GMR spin valve.

2.3. OOMMF simulation

Modeling the multidomain switching array was performed on OOMMF [79] using the 2D Periodic Boundary Condition extension [80]. Images from simulation are shown in **Fig. 4.2**. The magnetic parameters used for simulation were as follows: micromagnetic element length and width 10nm, element thickness 3nm, saturation magnetization 1×10^6 A/m (between CoFe and NiFe), uniaxial anisotropy constant K1 = 1×10^4 J/m^3 along x-axis,

exchange constant A = 3×10^{-11} J/m, and Gilbert damping constant 1.0. The simulation was done without considering thermal energy.

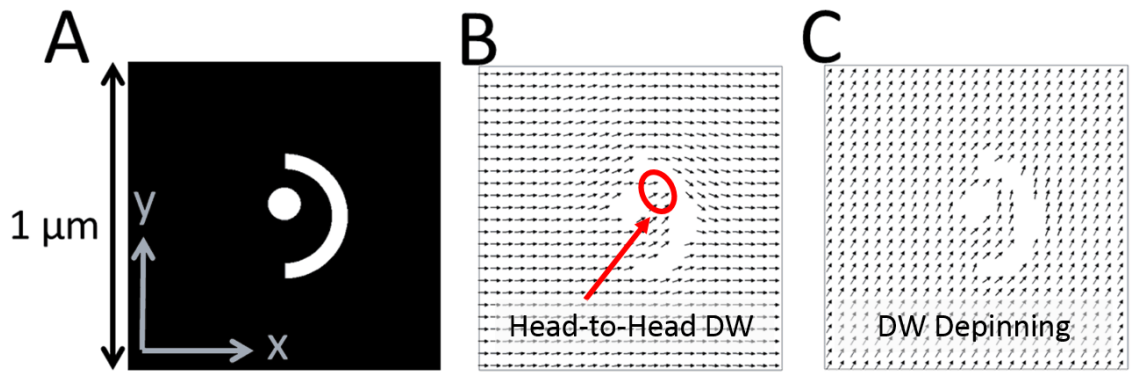

Fig. 4.2. A) Bitmap drawing as simulation input. Simulation tiles in an infinite 2D array of this pattern. Black represents magnetic material, and white represents empty space. B) Head-to-head domain wall formed during saturation is +x direction. The structure remains pinned at zero applied magnetic field. C) Magnetic switching is triggered by depinning of head-to-head wall during field application in -x direction.

A field sweep was applied starting with 500 Oe in the +x direction, reducing by 10 Oe with each step, and ending at 500 Oe in the -x direction. The micromagnetic vectors follow the dynamics of the Landau-Lifshitz-Gilbert equation (**Eq. 1.1**) and fully relax to equilibrium after each step of the field sweep. The condition for full relaxation is to stop the simulation after the maximum dm/dt found in any cell passes below a threshold of 0.01 degrees per ns.

Micromagnetic simulation showed that the multidomain switching array can be used to pin and de-pin an infinite 2D array of domain walls. The domain wall remained pinned while decreasing the initial magnetic field from +500 Oe to -10 Oe along the x axis. Depinning occurred between -10 Oe and -20 Oe.

3. Experimental results

Devices were patterned with electron beam lithography as shown in **Fig. 4.3**. To optimize the signal-to-noise ratio, we varied the feature size. Nominal feature sizes of

0.5µm, 0.75µm, 1.0µm, and 2.0µm were included in the fabrication, but only 0.5µm, 1.0µm, and 2.0µm were included in the MNP immobilization experiment. Optical microscope and SEM images after fabrication steps 4 and 5, respectively, are illustrated in **Fig. 4.4**.

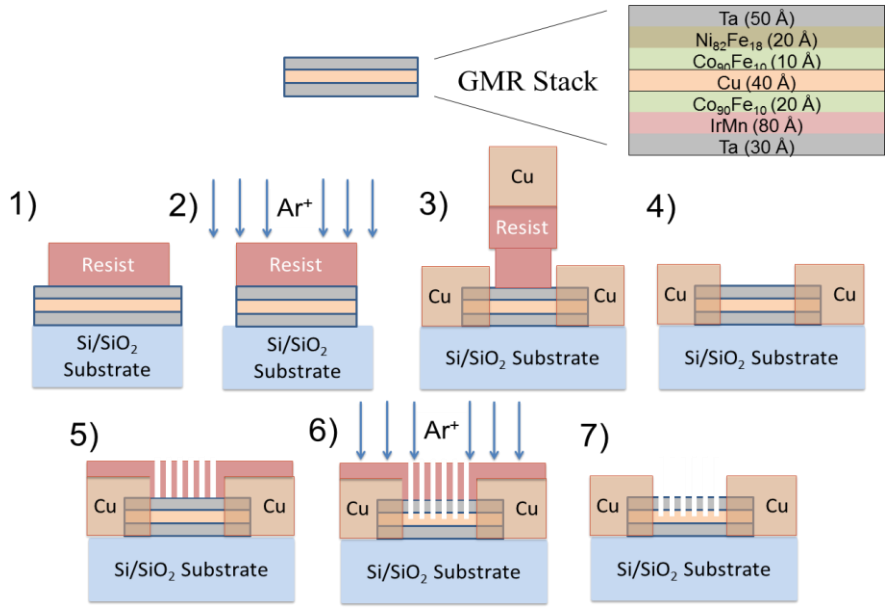

Fig. 4.3. Steps of Fabrication. 1) Photolithography, positive resist. 2) Ion milling, resist removal. 3) Photo-lithography, positive bi-layer resist. 4) Lift off deposited Cu to create electrodes. 5) E-beam lithography, positive resist. 6) Ion milling, resist removal. 7) Result.

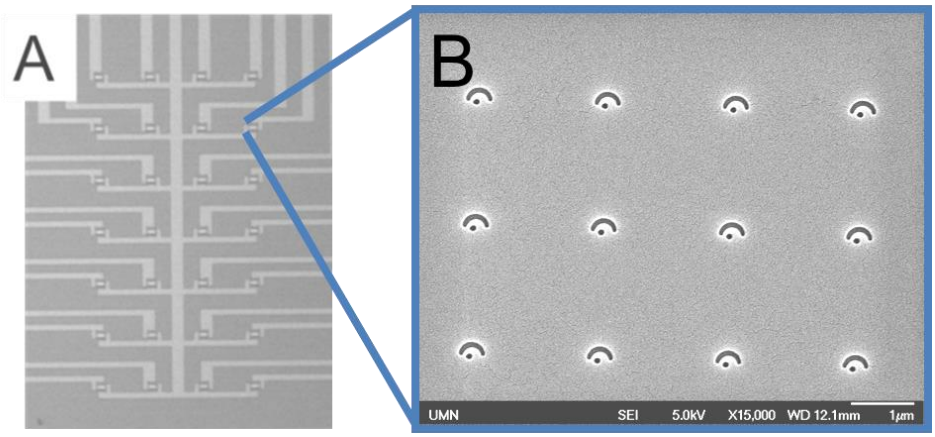

Fig. 4.4. A) SEM image of 28 multidomain switching array sensors. B) Detail of a small portion of a single multidomain switching array sensor.

Both AFM and MFM were performed to examine the patterned structures. Just prior to performing AFM and MFM experiments, we applied a magnetic field to saturate the magnetic free layer. From the MFM images, we can see some light and dark areas related to the stray magnetic field in those areas as illustrated in **Fig. 4.5**. Based on our knowledge of the saturation direction, we know that the pinned domain wall is head-to-head, so the dark areas must represent positive magnetic charge. Conversely, the light areas represent negative magnetic charge.

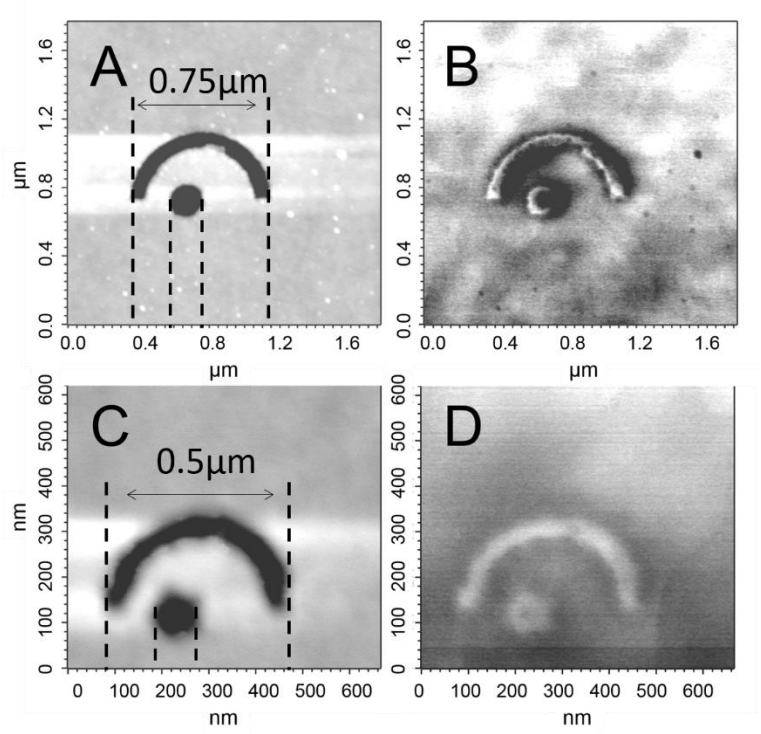

Fig. 4.5. Samples were saturated with applied magnetic field according to **Fig. 4.2** to validate simulation. A) AFM of nominally 0.75µm feature size. B) MFM of 0.75µm feature reveals magnetic stray fields from positive (dark color) and negative (light color) magnetic charge. From this data we can detect the presence of a head-to-head domain wall. C) AFM of nominally 0.5µm feature size. D) MFM of 0.5µm feature does not clearly reveal any pinned domain wall.

We found that we could indeed detect MNPs as illustrated in **Fig. 4.6**. The trend shows slightly higher signal-to-noise ratio as the feature size increased. More work is required,

however, to model the physics of domain wall depinning in the presence of MNPs. We have designed a new large area sensor with multidomain switching. From the transfer curve, we see that the switching is multidomain rather than coherent rotation.

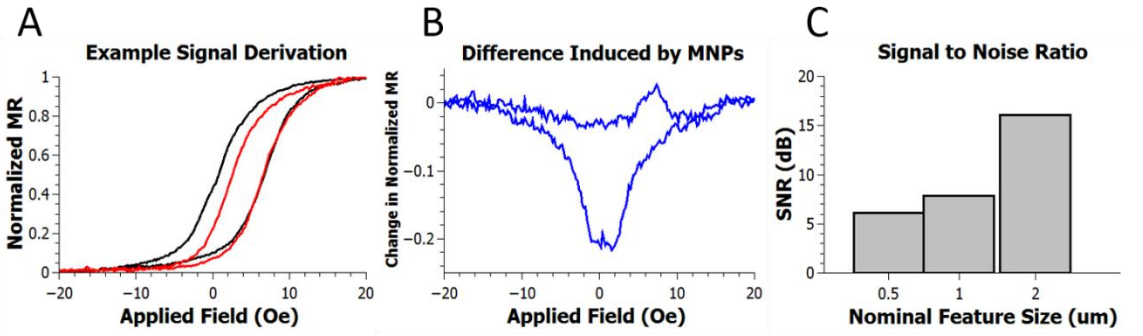

Fig. 4.6. A) Black curve collected prior to MNP deposition is used as a background signal. Noise derived from several consecutive measurements. Red curve collected after MNP deposition. B) Signal derived from difference between red and black curves in A. C) Signal to noise ratio improved with size of feature as measured in **Fig. 4.5**.

4. Discussion

4.1. Large area sensor with multidomain switching advantages

The large area sensor with multidomain switching addresses sensitivity limitations by orienting the applied field nearly parallel to the easy axis defined by induced anisotropy during film deposition. The sensitivity of this type of sensor from our group was 0.59 % / Oe [33] as opposed to the best linear sensors used in this study with 0.07% / Oe. In addition to the increased sensitivity, the large area sensor with multidomain switching is simple to fabricate and does not have any gaps in the sensing area. Although we cannot yet say for sure that the MNP position does not matter, it seems likely that this is the case.

4.2. Limitations of large area multidomain sensor

After this work was performed, we continued the theoretical investigation of domain nucleation switching in the large area sensor [63]. This updated model provides a much

better understanding of the outstanding performance of the large area biosensor when used with high-moment magnetic nanoparticles.

Increased sensitivity comes at the cost of added complexity in signal processing. The two sides of the hysteresis loop must be treated separately, and the easiest way to manage this is by direct subtraction of the full hysteresis loop from a baseline loop on each sensor prior to running the assay. This procedure does not lend itself to frequency-domain signal processing. In addition, the theoretical model for this type of sensor is descriptive but not quantitative in the sense that simulations can predict the signal.

Chapter 5. Domain Wall Sensor

1. Introduction

As indicated in the previous chapter, we were motivated to explain the extraordinary results from past work on the large area sensor with high-moment CoFe MNPs [19, 33, 34]. We recognized early on that the conventional model in the literature for MNP detection was based on a uniform free layer with coherent rotation. This was clearly not the case in our large area sensor, so we focused our attention on the impact of domains and domain walls. We also recognized that the large area sensor did not give us direct control to manipulate domain walls, so we turned our attention to structures that would. In the process, we devised a theoretical model and an experiment to help observe the impact of domain walls in GMR biosensors.

Previous literature indicated that MNPs could be detected through their interaction with domain walls in AMR thin films [81]. In the cited work, a domain wall was used to detect a single bead filled with magnetic nanoparticles. Sensitivity at this level, if used properly, would allow for the counting of magnetic labels in an ovarian cancer assay. Our job then became determining a possible configuration of a magnetic biosensor based on domain walls.

Inspired by this work, we wanted to devise a system that would be compatible with a CMOS array so that many domain wall sensors could form an array with enough surface area coverage to form a biosensor. If the domain wall is contained within the free layer of a GMR sensor, then the domain wall location can be determined from the resistance of the sensor itself. This type of system would be compatible with a CMOS array, so the positions of thousands or even millions of domain walls would theoretically be accessible. We hoped

to see an interaction between magnetic beads and the GMR free-layer domain wall that allows for quantification of biomarkers.

This sensing scheme has many practical challenges in terms of fabrication, but we chose to pursue a very modest version that involved detecting magnetic nanoparticles in a single location in the center of the patterned track within a GMR film. Additionally, we decided to present an analytical model to help understand basic questions such as the importance of magnetic nanoparticle height, width of sensor track, position of the MNP, and the effect of an external applied magnetic field on the detection.

2. Theoretical model

2.1. Goal of analytical model

The interaction between domain walls and superparamagnetic particles is known to result in an attractive potential. The goal of analysis is to quantify this potential for sensor optimization. The calculated change in depinning field (ΔH_{dp}) was compared against previous experimental data from the literature and micromagnetic simulation of the same experiment. These comparisons provide justification for further experiments with the sensing scheme.

The domain wall model is illustrated through potential energy and effective field induced by the presence of a single superparamagnetic particle above a magnetic domain wall in a 5nm ferromagnetic film (M_s = 800 emu/cm^3) with uniaxial crystalline anisotropy ($K_u < 10^7$ erg/cm^3). The wall width, wall type (head-to-head, Néel, and perpendicular Bloch), film dimensions, particle height, and external applied field affect the performance of particle sensors. The analytical model can be used to guide future experimental methods.

2.2. Weak pinning potential

Interactions between superparamagnetic particles and several types of domain walls can be characterized in terms of potential wells. In sensing applications, these wells constitute extrinsic pinning sites that equate to the sampling of a local stray field from each particle. In general, a narrow domain wall will produce larger values of ΔH_{dp}. In the case of the Néel wall on a 2 μm by 1 μm film, the optimal width is approximately 1.2 times the particle height as seen in **Table 5.1**. This optimal wall width can increase the value of ΔH_{dp} so that in some cases it is more than double the narrow-wall (high anisotropy) case.

Table 5.1. Change in depinning field (ΔH_{dp}) induced by a particle at various heights (z_p) is fitted to a power law function so that it can be compared to the $H_0/|r|^3$ dipole field strength where A and H_0 are proportionality constants. The power (B) is seen to decrease with increased wall width. *[57]*

Wall:	**Head-Head**	**Néel**	**Perpendicular**
δ (nm)	**B**	**B**	**B**
10			2.39
20			2.12
40	1.69	2.59	1.91
100	1.24	2.03	
200	0.87	1.77	$\Delta H_{dp} = A/z_p^B$
400	0.61	0.22	

This unique interaction can be understood by close inspection of **Fig. 5.1.** In all three cases shown, the magnetostatic symmetry is such that the lowest energy state occurs when the particle is located directly above the wall center. In the cases of the head-to-head and perpendicular Bloch wall, the alignment of film magnetization and particle stray field becomes more pronounced in regions away from the wall center. This is in stark contrast to the Néel wall in which the reverse is true. This fundamental difference manifests itself in a quantifiable way when the potential energy of particle-wall interactions is analyzed.

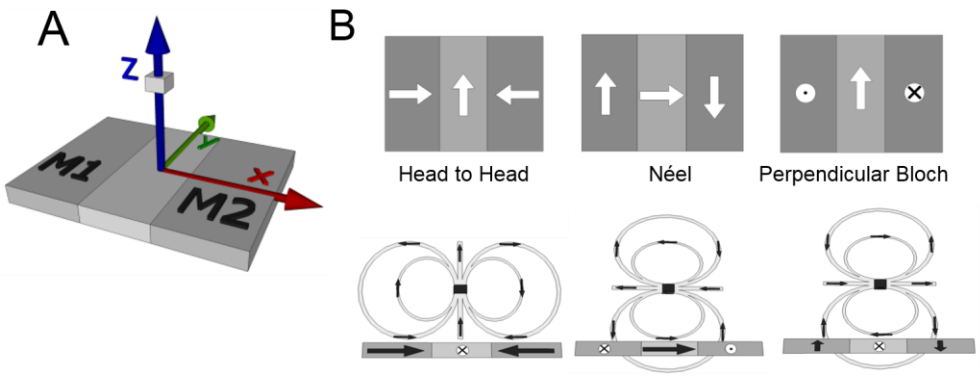

Fig. 5.1. A) Schematic of the particle and sensor and B) types of domain walls studied. [57]

2.3. Description of weak pinning

The strategy employed was based on the construction of potential energy plots for the domain wall types shown in **Fig. 5.1.** The relative position between the particle and wall center along the x-direction causes a change in potential energy resulting from magnetostatic interaction between the particle and domain wall. The particle is assumed to remain in a fixed position with the wall moving beneath as would usually be the case for sensing applications.

2.4. Model derivation

The following equations were used in the quantitative description of weak domain wall pinning in the presence of MNPs. The interaction begins with superparamagnetic particle excitation due to the domain wall stray field ($\mathbf{H_w}$):

$$\mathrm{H_w} = -\int \mathrm{d^3r'}\nabla\cdot\mathrm{M}\frac{(\mathrm{r}-\mathrm{r'})}{\|\mathrm{r}-\mathrm{r'}\|^3} + \int \mathrm{d^2r'M}\cdot\hat{\mathrm{n}}\frac{(\mathrm{r}-\mathrm{r'})}{\|\mathrm{r}-\mathrm{r'}\|^3} \qquad \textbf{Eq. 5.1}$$

The particle is thus subjected to a total field ($\mathbf{H_t}$) including any external applied field ($\mathbf{H_a}$) and obtains a moment according to the Langevin function with temperature set to 300 K:

$$\langle \mathrm{m_p}\rangle = \mathrm{m_0}\left(\coth\frac{\mathrm{m_0 H_t}}{\mathrm{k_B T}} - \frac{\mathrm{k_B T}}{\mathrm{m_0 H_t}}\right)\hat{\mathrm{H}}_\mathrm{t} \qquad \textbf{Eq. 5.2}$$

The particle then produces a stray field of its own ($\mathbf{H_p}$). The simulated particles in this case are cubes of $Fe_{70}Co_{30}$ with side length of 10 nm and saturation moment (m_0) of 2×10^{-15} emu. Due to such a small size, the particles are well approximated as point dipoles in this context:

$$\left\langle \vec{H}_p \right\rangle = \frac{1}{r^3}\left[3\hat{r}\left(\left\langle \vec{m}_p \right\rangle \cdot \hat{r}\right) - \left\langle \vec{m}_p \right\rangle\right] \qquad \textbf{Eq. 5.3}$$

This field can then be used to calculate the change in potential energy (ΔU) due to the existence of a particle:

$$\left\langle \Delta U \right\rangle = -\int d^3\vec{r}\,'\vec{M} \cdot \left\langle \vec{H}_p \right\rangle \qquad \textbf{Eq. 5.4}$$

Following this approach, ΔU is calculated for a sequence of wall positions, and a local minimum in energy (potential well) is observed when the particle is located directly above the wall center.

The changing moment ($\mathbf{m}_f$) of the film allows us to model the potential well as an effective field ($\mathbf{H}_{eff}$):

$$H_{eff} = -\partial(\Delta U)/\partial m_f \qquad \textbf{Eq. 5.5}$$

Only the change in potential energy (ΔU) due to a particle is calculated, and so the negated effective field represents the change in depinning field ($\mathbf{\Delta H_{dp}}$) because of the particle. This change can then be compared with experimental data in sufficiently simple circumstances.

2.5. Height of MNPs

One valuable piece of information for magnetic nanoparticle sensor designers is the relationship between particle height and signal strength. This relationship can be seen to change with both sensor aspect ratio and domain wall width. The results from several wall

varieties are summarized in **Table 5.1**. ΔH_{dp} is found to be inversely proportional to the particle height (z_p) raised to some power (B) which decreases as wall width increases. This finding corroborates the idea that narrow domain walls sample local particle stray fields. As wall width increases, more magnetostatic alignment occurs during integration in **Eq. 5.4**. Increased averaging creates less dependence on the maximum local stray field from a particle and therefore decreases the value of B.

Table 5.2. ΔH_{dp} can be optimized by tuning the Néel wall width for strongest magnetostatic interaction with the particle at each specified particle height (z_p). The value of optimal ΔH_{dp} is shown in proportion to that for the 40 nm (high anisotropy) wall width. [57]

	Optimal Néel Wall	
$\mathbf{z_p}$ (nm)	$\boldsymbol{\delta_{opt}}$ (nm)	$\boldsymbol{\Delta H_{opt}/\Delta H_{40}}$
80	95	1.44
100	120	1.98
120	145	2.82

In the cases of head-to-head and perpendicular configurations, ΔH_{dp} increases monotonically with decreased wall width. This trend is just as one would expect in terms of creating a high stray field to excite the particle. The trend for Néel walls is subtler and is therefore presented in **Table 5.2** for a 2 μm by 1 μm film. Our model predicts that there is an optimum wall width and that decreasing past this value causes a decrease in ΔH_{dp}. The importance of this trend can be seen by comparison between ΔH_{dp} for optimal wall width versus that for the 40 nm wall.

Inspection of **Fig. 5.1** shows that the Néel wall is unique in that the particle stray field aligns most closely with the interior portion of the wall. If this interior region becomes arbitrarily small, then any anticipated gain resulting from greater H_w in **Eq. 5.1** is offset by the loss of magnetostatic alignment in **Eq. 5.4**. In addition, we find that ΔH_{dp} is roughly proportional to the derivative of H_w with respect to wall position in the limit of narrow

Néel wall and low applied field. We find also that the value of this derivative peaks at a wall width is very near the optimal wall width (δ_{opt}) but begins to underestimate δ_{opt} as particle height increases. This happens because δ_{opt} increases with particle height and subsequently enhances the dependence on volume of magnetostatic alignment as described above.

2.6 Sensor width variation

Another design consideration for sensor optimization is the overall size and shape of the film. Even though we are trying to model a wide range of films, we must be careful to maintain reasonable bounds. Due to shape anisotropy considerations, the head-to-head film should be very narrow in comparison to the Néel film, whereas the perpendicular film is relatively impervious to this parameter.

As can be seen in **Fig. 5.2(A-C)**, all three wall types seem to have distinctly different responses to variation in film width. The most obvious feature is that the perpendicular Bloch wall gains a repulsive potential at small film widths. We also find that, in the hypothetical situation where the Néel wall can survive on narrow films, it too would gain a repulsive potential. The biggest difference between these potentials then becomes the proportion of change on the edges versus that in the center of each well. We find that all three trends can

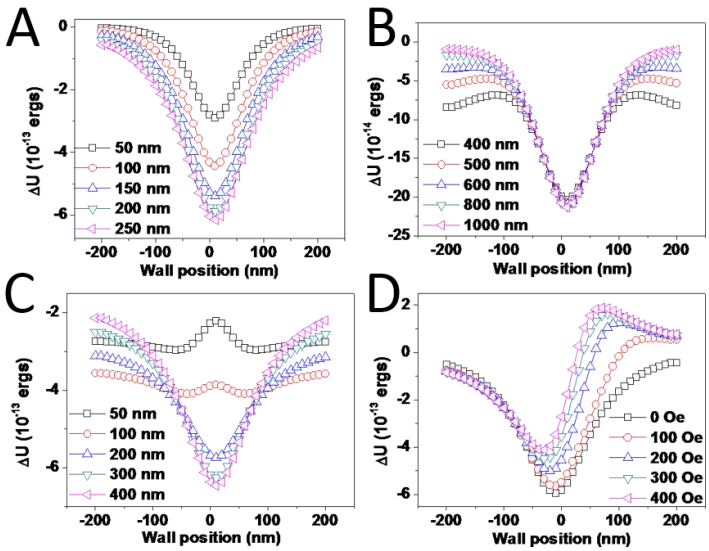

Figure 5.2. ΔU vs. x plots change shape depends on film width for A) head-to-head, B) Néel, and C) perpendicular Bloch walls. D) ΔU symmetry depends on applied field. [57]

be explained by the fact that the particle moment gains a y-component as the film becomes narrower.

2.7. MNP location variation

With increased y-component excitation, the particle stray field tilts so that it loses net alignment with film magnetization, which explains the decreasing depth of head-to-head and perpendicular Bloch potential wells. In contrast, the Néel potential gains a local maximum on either side of the well while the center remains relatively fixed as films narrow. At film widths less than those shown, the two local maxima grow and move inward until they coalesce in the center. This can be understood by the fact that the y-component of the particle moment is antisymmetric in this case. When the particle is directly above the wall center, the moment has zero y-component. In regions far outside the Néel wall, a strong y-component induces net magnetostatic alignment where there otherwise would be none. It is only the intermediate regions that suffer losses like those found in the other two structures.

2.8. External applied field

One final consideration is the application of additional fields either through an external source or simply due to Oersted fields during measurements. If the applied field coincides with an antisymmetric component of wall stray field, then an asymmetry will be observed in the well as shown in **Fig 5.2D**. This type of asymmetry can be important when operating conditions involve large applied fields [81].

3. Experimental results

3.1. Problem specification

For preliminary optimization, the film was taken to be an unspecified material with saturation magnetization (M_s) of 800 emu/cm^3 and uniaxial crystalline anisotropy (K_u) up to 10^7 erg/cm^3 for the perpendicular case and up to 5×10^5 erg/cm^3 for the in-plane cases. This range of values for K_u allowed us to vary the wall width from 10 nm to 50 nm for the perpendicular case and from 40 nm to 400 nm for the in-plane cases. We define the magnetization configuration according to the analytical equation for a Bloch Wall in bulk material to define a domain wall width (δ) [59]:

$$\theta(x) = \tan^{-1}\left[\sinh\frac{\pi x}{\delta}\right] + \frac{\pi}{2} \qquad \textbf{Eq. 5.6}$$

3.2. Comparison to the literature

Experimental depinning measurements [81] for superparamagnetic bead detection were used to validate our model. The experiment involved placing four 80 nm Micromod Nanomag-D beads at the corner of a square Permalloy AMR split-ring. The particles were initially attracted to a head-to-head wall residing at the corner and then created an extrinsic pinning potential for the wall such that the depinning field was 262 Oe instead of the usual 250 Oe without the particle. Micromagnetic simulation was cited as agreeing well with this value [81].

To accomplish the corresponding energy calculation in our framework, two analytical simulations were used to set upper and lower bounds on the result. In both cases, we began the simulation with a head-to-head domain wall along the 45-degree diagonal from the top left corner of a 1 μm by 180 nm Permalloy strip. The wall width was set equal to the strip width. In the first case, we treated the dynamics as a discrete two-stage process with an

initial 90-degree head-to-head wall and a final 90-degree Néel wall. In the second case, we approximated the dynamic situation with a 180-degree head-to-head wall that rotated as a rigid unit with the point in the corner remaining fixed. During both simulations, a constant field of 250 Oe was applied in the x-direction. Atomic force microscope images from the cited experiment were used to estimate particle positions. Each particle was approximated to be a point dipole with a moment of 4.5×10^{-14} emu located 70 nm above the surface of the 30 nm-thick Permalloy film.

An application of our potential well calculation using the analytical magnetization states described above yielded a +3.4 Oe change in depinning field for the first case and +22.0 Oe change in depinning field for the second case. Since the quoted value of +12 Oe lies near the center of our bounds, we were confident that the method would perform well upon integration with micromagnetics. Furthermore, the doubling of ΔH_{dp} with decreased strip width from 180 nm to 80 nm is predicted by micromagnetic simulation [81]. This result is in very good agreement with our prediction of increase by a factor of between 1.3 and 3.4 under the same circumstances. With these results in mind, we limited our conclusions to those involving trends rather than absolute magnitudes. Also, our results were limited to cases with the particle positioned directly above the center of the film.

As outlined above, DWs provide means of interaction between MNPs and thin magnetic films. As such, they are a basic tool for studying this interaction and its applications [58]. One characteristic of DWs is that they are easily pinned by defects (called pinning sites), both fabricated and intrinsic. The applied field required to release the DW from such a pinning site is often called the "depinning field." Change in depinning field strength due to the presence of magnetic particles near a NiFe nanostrip has been

demonstrated in the literature [81]. This change in depinning field strength provides justification for the claim of strong interaction between MNPs and thin magnetic films.

The sensing of MNPs for biological applications presents a unique set of design criteria which differ from other field-sensing applications. The size of sensing structures is generally optimized by matching the size of MNPs [82]. Furthermore, quantifiable detection of biological molecules is simplified by one-to-one tagging between MNPs and biological molecules. For this reason, sensing structures and MNPs are expected to have best performance if their size is on the same scale of the targeted biological molecules. DW widths can be scaled down to tens of nanometers or less with the appropriate choice of materials and structures [83]. This makes them an attractive candidate for MNP sensing structures.

3.3. Sensing MNPs with GMR domain wall

Head-to-head DWs were formed and detected in U-shaped spin valve nanostrips by a Bitter Method (see below) on a scanning electron microscope. We found that optimization of the washing technique was critical to proper deposition of MNPs. The DW position was monitored electrically with a GMR. DWs were repeatedly pinned and depinned from a site near the device center using a two axis-field (x and y directions in **Fig. 5.3A**). We first confirmed the distribution of depinning events from a random sample of devices and obtained a standard deviation of 2.2 Oe. We then obtained an experimental signal of 8 Oe shift in depinning field from just 7 MNPs of 35nm diameter.

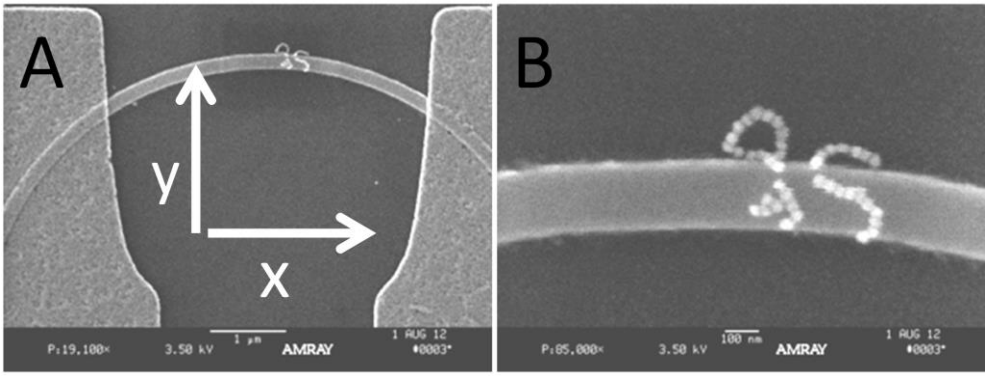

Fig. 5.3. A) The Bitter Method with SEM was performed to confirm the existence of a single domain wall near the center of a curved structure. The MNPs used were 35 nm diameter iron oxide coated with PEG and diluted in DI water. Surfaces were rinsed with clean deionized water before imaging. B) A high magnification image of the same device shows the chains of individual MNPs. *[58]*

It is well-known that MNPs are attracted to DWs through magnetostatic interactions. In short, the DW stray field excites an MNP, and this MNP is subsequently drawn to nearby regions of high field gradient. The application of this concept is commonly referred to as the "Bitter Method" when used to visualize DWs. In certain cases, we may expect an advantage from using such magnetostatic interactions to concentrate and/or manipulate MNPs in biological applications [27, 84, 85, 86]. On the other hand, we wanted to eliminate the effect so that it did not interfere with biologically specific bonding. For this reason, we preferred structures where domain walls could be easily created and annihilated. Furthermore, it was assumed that biologically specific bonding events may occur in any location on the sensor. Therefore, it was also preferred to have some control over the starting position of the created DW. Based on these criteria, we selected half-ring structures as shown in **Fig. 5.3**.

Based on electrical measurements, we found that a momentary application of 250 Oe along the y-axis could be used to reliably create a domain wall near the center of the device. Similarly, a momentary application of 250 Oe along the x-axis could be used to reliably

reset the device to its original condition. We expected that the application of a field at some intermediate angle would create a wall at some position away from the device center. Future experiments may be used to confirm this.

GMR spin valve films were deposited at the University of Minnesota with a Shamrock Magnetron Sputter System onto Si / SiO2(1000) substrate: Ta(30) / IrMn(80) / Co90Fe10(20) / Cu(40) / Co90Fe10(10) / Ni82Fe18(50) / Ta(50), with thicknesses in parentheses given in Angstroms. Curved spin valve nanostrips were then fabricated by electron-beam lithography and subsequent ion milling followed by electron-beam lithography and liftoff of Cu contacts. Fabricated wire widths of 100 nm and 200 nm were confirmed by JOEL 6700 SEM. Both steps of electron-beam lithography were performed on a Vistec EBPG5000+ 100 KeV system. Fabricated samples were annealed at 200°C for 60 minutes and then allowed to cool back to 30°C. The entire annealing process was carried out in a field of 4kOe along the x-axis (**Fig. 5.3A**). The GMR of fabricated devices was found to be 2.5% (**Fig. 5.4B**).

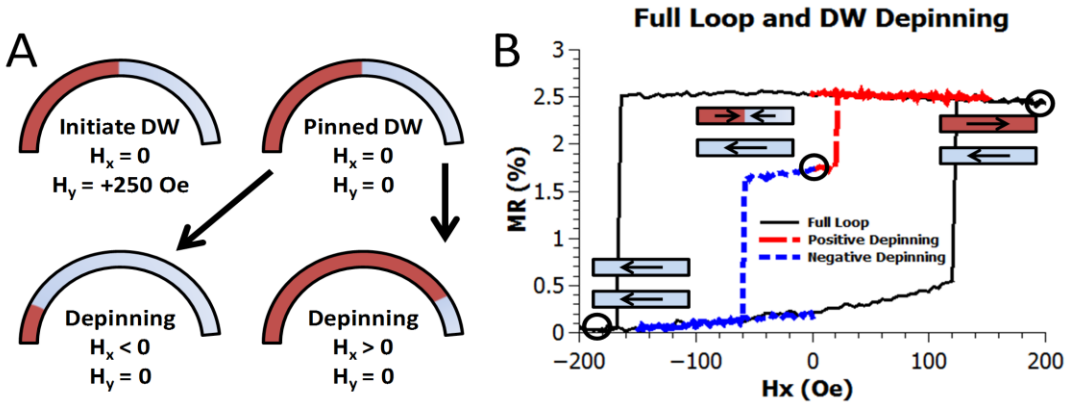

Fig. 5.4. A) Top view schematic of the free layer during the demonstration of domain wall pinning and depinning in a GMR spin valve (clockwise from top left): Initiate DW, Pinned DW, Positive Depinning, Negative Depinning. B) Side view schematic of GMR layers during the demonstration of domain wall pinning and depinning. *[58]*

Setting of domain walls in the curves was accomplished by momentary application of a field along the y-axis. Field-induced DW depinning was measured electrically by GMR

during the application of a field along the x-axis (**Fig. 5.4)**. The stage used for electrical testing included two perpendicular pairs of electromagnets for simultaneous application of x- and y-directed magnetic field components each with -250 Oe to +250 Oe range.

3.4. MNP deposition

The Bitter Method relies on the attraction of MNPs to locations of stray field gradient to observe DWs on the surface of magnetic films. If a fluid containing well-dispersed MNPs is brought in contact with the magnetic film, then the DWs will collect MNPs, which can later be viewed with a microscope. In this case, the goal was to reliably deposit a small number of MNPs onto a pinned domain wall so that its position may be observed with SEM. The MNPs were 35 nm Fe_3O_4 with polyethylene glycol coating purchased from Ocean Nanotech. The MNPs were suspended in water with good colloidal stability, so any clustering was assumed to be from the DW stray field.

The Bitter Method for domain wall observation was composed of five steps: 1) DW setting with momentary application of Hy = +250 Oe; 2) MNP deposition (10 ng/mL); 3) MNP incubation (3 hours, 97% humidity); 4) Washing / drying, and 5) SEM observation. Optimization of step 4 proved to be very critical, and several techniques were attempted. The most successful washing / drying routine was also the simplest: after the MNP incubation period, the sample was immediately placed under a stream of deionized water for 3-5 seconds. After that, the sample was gently dried with N_2. This washing technique gave the best balance of clean substrate overall and 1-30 MNPs bound to each single DW area. In contrast, some techniques removed nearly all MNPs, while others left a density of MNPs on the order of 400/μm^2 in the field. As a point of reference, a rectangular 2D arrangement of 35 nm particles would give a density of 784/μm^2.

According to previous literature regarding the dimensions of magnetic nanostrips [87], the head-to-head DWs in all of our devices can be assumed transverse as opposed to vortex. This assumption implies a simple DW position detection scheme based on GMR spin valve nanostrips. Since the fixed layer magnetization was set along the x-axis, there was an approximately linear relationship between DW position and observed magnetoresistance. This becomes a close approximation near the region of interest as indicated by the Bitter Method result.

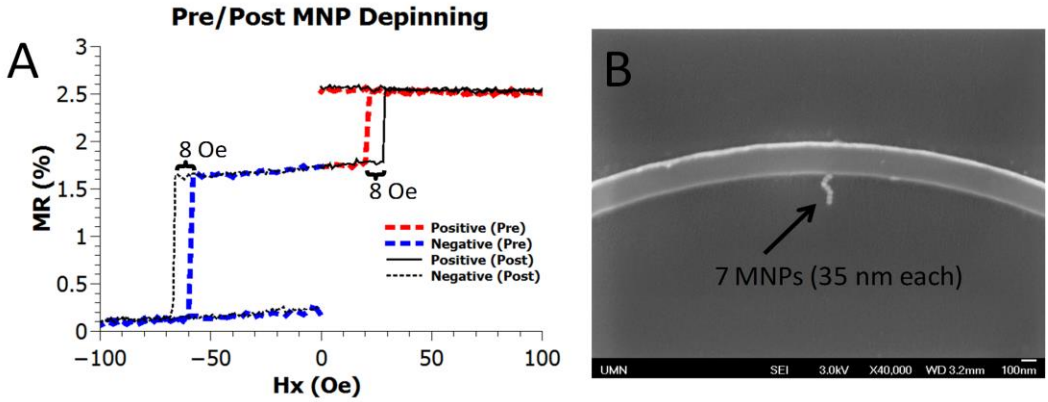

Fig. 5.5. A) Results from one device before and after application of 7 MNPs with the Bitter Method. This device shows a change of 8+/-1 Oe in depinning field for both forward and reverse directions. B) SEM image of the device after Bitter Method shows 7 MNPs that give rise to the change in depinning. *[58]*

The electrical detection of domain wall depinning consisted of a sequence of five steps: 1) Initial GMR loop; 2) DW setting with momentary application of Hy = 250 Oe; 3) Negative field depinning; 4) DW setting with momentary application of Hy = 250 Oe; 5) Positive field depinning. The steps are illustrated in **Fig. 5.5A**. We found that no notches were required to hold a domain wall near the center of the device.

The pinning sites can be attributed to defects from patterning, deposition, or both. In many cases, we observed multiple pinning sites as one would expect from random defects. Even so, we found that the most dominant depinning site of each individual device was

repeatable with standard deviation of 2.2 Oe. In one particularly stable device, we tested the forward and reverse depinning fields both before and after MNP deposition by Bitter Method. The result, illustrated in **Fig. 5.5A**, shows an 8 Oe shift due to deposition of 7 MNPs of 35 nm diameter (**Fig. 5.5B**).

To quantify the signal, it was important to consider the distribution of depinning repeatability as noted above. For this, we used the first 100 measurements spanning 11 devices (**Fig. 5.6**). We subtracted away the systematic variation from one device to the next so that we could focus on just the random variation of each device. To determine the statistical significance of the signal, we performed a Wilcoxon Signed Rank Test, a non-parametric test appropriate for our small sample size (N = 6 pairs), using SAS software version 9.3 (Cary, NC). We calculated a p-value of 0.0313, indicating the signal due to MNP deposition was statistically significant from 0 at a 0.05 significance level. The reported result clearly illustrates a strong interaction between MNPs and DWs.

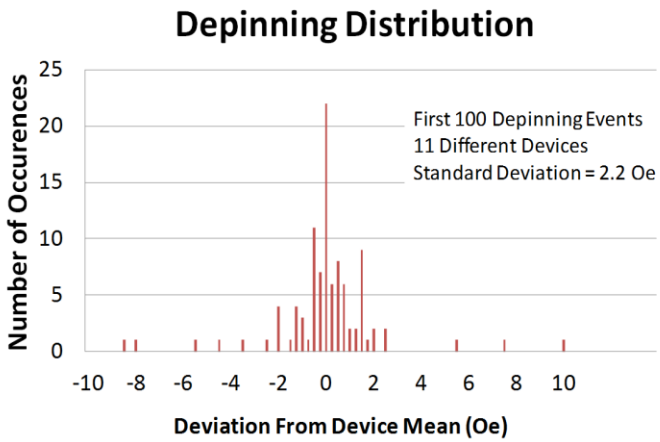

Fig. 5.6. Histogram representing the variation in depinning field away from device means. *[58]*

4. Discussion

4.1. Domain wall sensor advantages

We have demonstrated the technique for proper deposition of MNPs as well as statistical summary of the depinning measurements. In this way, we promote the use of head-to-head domain wall as a promising structure for detection of a small number of MNPs. The domain wall sensor has the capability of detecting several MNPs. With such

high sensitivity, this type of sensor has the potential to act as a digital count of magnetic beads.

4.2. Domain wall sensor limitations

The motivation for studying the domain wall MNP sensor was to validate the possibility that domain wall pinning could help explain the large area sensor. Indeed, we can say that there was a strong interaction between domain walls and MNPs in this experiment. In the case that we studied, the small number of MNPs produced a relatively large signal. It is tempting to use this as evidence of MNP-induced domain wall pinning within the large area sensor; however, we must be careful to distinguish the head-to-head domain wall from the Néel walls that most likely form in the large area sensor. The walls in the large area sensor do not produce a large magnetic field compare to the case of the head-to-head wall in this experiment.

The main disadvantage to the domain wall sensor is the difficulty in scanning a large surface area for MNPs. Without the use of on-chip CMOS, we cannot expect to use this design for normal biosensing applications. Also, it is not entirely clear whether the domain wall sensor can have a higher sensitivity than a linear sensor of the same size. More work should be done to clarify this comparison.

One more technical challenge for this design is the requirement of two perpendicular magnetic field sources in the same (x-y) plane as the GMR. Finally, it should be noted that there is a possibility of stray fields from the domain wall sensor causing non-specific binding due to large gradient and consequently large attractive force between MNPs and the head-to-head domain wall.

Chapter 6. zLab System Integration

We developed both a bench-top and a hand-held GMR biosensing system that can contribute to the development of ovarian cancer assays and serve as a platform for detecting a wide variety of other biomarkers. The bench-top and hand-held systems use identical PCB, firmware, and software. The differences between the two systems are superficial except for the number of sensors available (64 in the case of bench-top and 29 in the case of hand-held). System integration, led by the author, included spintronics and magnetic materials engineering, design of the magnetic coil with a ferrite core, electrical engineering, mechanical engineering, analog and digital signal processing, firmware programming, user interface programming on both a PC and an Android smartphone with communications over both USB and Bluetooth. Development of both systems and the use of the bench-top system on a highly sensitive multiplex detection of ovarian cancer biomarkers (chapter 7) is our main contribution to the field of GMR biosensing.

1. Introduction

The system used for GMR biosensing was not originally dedicated hardware. It consisted of a Kepco BOP 20-20 bipolar power supply running a large pair of coils, a Keithley 6221 current source, a National Instruments data acquisition system with Labview program to interface both the Kepco and Keithley instruments with a computer, and a set of probes on micropositioners to contact the pads of a GMR biosensor while looking through a stereo microscope. These laboratory instruments were shared with group members working on other projects, and so we often needed to reassemble or maintain the system in some way. In addition to our desire for convenience and efficiency, we also wanted to create a system that was self-contained and portable so that future students could

work more easily with outside partners such as the University of Minnesota medical school and the Mayo Clinic.

After several iterations of initial development, we found help from a group of senior undergraduate students who, as a capstone project for their degrees, followed our specifications to lay out and populate a PCB in addition to writing firmware and software to control it through a convenient user interface. The result was zLab, a handheld GMR biosensing system. The success of this design project became a catalyst to assemble a team, refine our system, and submit the work to the 2nd round of the Nokia Sensing XCHALLENGE competition in 2014. According to the competition overview found on the official website [88]: "The Nokia Sensing XCHALLENGE is a $2.25 million global competition to accelerate the availability of hardware sensors and software sensing technology that individuals use to access, understand, and improve their health and well-being. Innovation in sensing is an important component to creating a means for appealing, usable, smarter digital health solutions."

This chapter describes the GMR biosensing system components developed by the Golden Gopher Magnetic Biosensing team, of which the author was student team leader. See Appendix A for a list of all team members. Our team received one of five Distinguished Awards which served as runners up to the grand prize winner [89].

2. Spintronics and magnetic materials

As described in chapter 2, we deposited GMR spin valve film, lithographically patterned sensors, and characterized performance both before and after patterning. In the case of zLab, we elected to use the linear sensor due not only to its straightforward electronic implementation, but also because the sputtered GMR spin valve films had relatively low coercivity compared to what we expected from previous experience.

In addition to characterizing the sensors, we also characterized the superparamagnetic nanoparticles to confirm high moment at low field. As shown in **Fig. 2.5**,[ML1] we considered magnetic beads from MACS (50nm diameter) and Ademtec (200nm diameter). Both beads are useful in different applications, and we must note the superior low-field slope of MACS beads. However, the moment per bead and therefore moment per biomolecular binding event is greater for the 200nm Ademtec bead due to its greater number of MNPs per bead. Consequently, the 200nm Ademtec bead was preferred for low limit of detection.

3. Electronic circuit

2.1. GMR chip and breakout PCB

Initially, the layout of our sensor chip was designed to interface with a 30-pin connector from smart phones. This allowed us to directly plug our GMR sensors into a socket and avoid conventional integration steps such as wire bonding. Meanwhile, we tried to maximize the use of space by including two sample sites, called "reaction wells," on each 16-mm chip. Each chip is therefore labeled during fabrication with "A" and "B" designating opposite sides of the reaction well (**Fig. 6.1A**). Since one of the thirty pins is

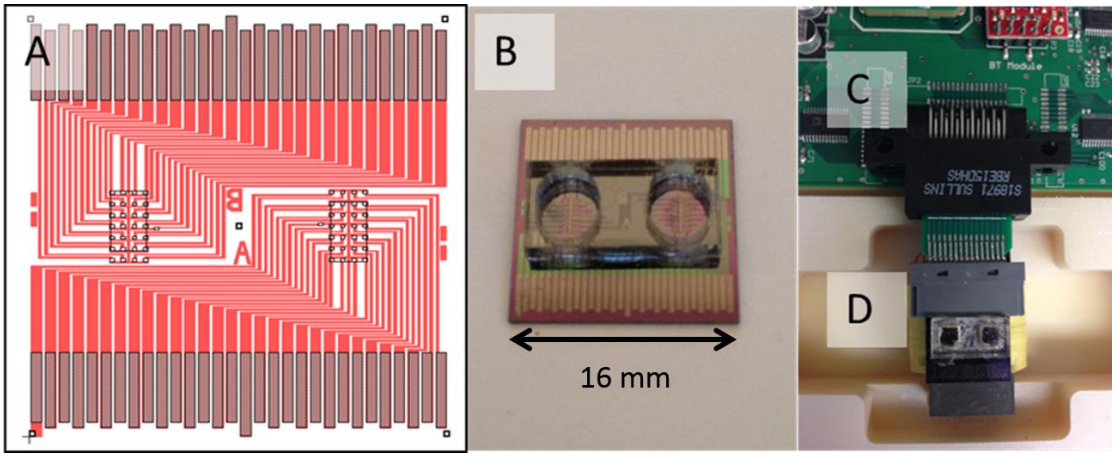

Fig. 6.1. A) GMR Biochip Layout. B) GMR Biochip after fabrication, annealing, activation, and reaction well. C) Socket for breakout printed circuit board (PCB) to connect to the main PCB. D) Socket for GMR Biochip to connect to the breakout PCB.

common to all sensors, we had room for 29 sensors on each side. Five sensors on each side were buried beneath relatively thick passivation so that they could serve as fixed references. The other 24 sensors remained exposed to the test sample.

2.2. Electrical circuit and signal generation

A simplified schematic of the electrical circuit is illustrated in **Fig 6.2**. Channel A of a 24-bit stereo audio CODEC drove a frequency f_1 at the top of a nearly-balanced Wheatstone Bridge circuit with a GMR biosensor in one of the four resistor locations. Each sensor was addressed through an analog switch multiplexer. Prior to testing, all the sensors were individually balanced against a resistance ladder. This ensured that the electronics were flexible enough to work with a range of sensor resistances even on a single biosensor chip. Channel B of the CODEC was routed to a voltage-controlled current source that drove a coil at frequency f_2.

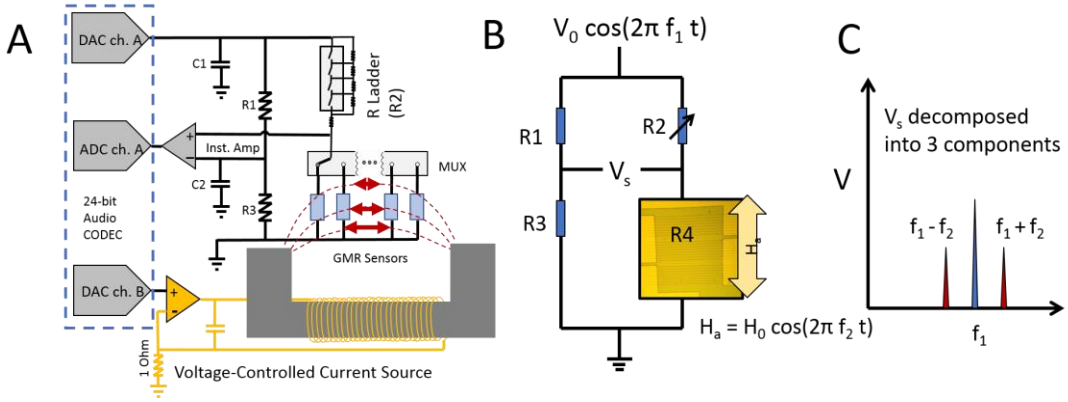

Fig. 6.2. A) Schematic representation of Wheatsone Bridge circuit on the multiplexing board *[52]*. B) Simplified schematic of Wheatstone Bridge C) Signal decomposed into frequency components. The two side tones (ω_1 +/- ω_2) represent the signals that change in response to MNPs.

The bridge drive frequency f_1 mixed with the coil drove frequency f_2 due to resistance modulation in the GMR on the sensing side of the bridge. The three resulting signals at frequencies f_1 (center tone) and f_1+/-f_2 (side tones) were balanced against two reference resistors on the other side. Appropriate balance was achieved by tuning the resistance

ladder for each individual sensor prior to the measurement. Since the reference resistors did not modulate frequency, they only produced a signal at the center tone. Subtraction and amplification of Wheatstone Bridge voltages occurred at the instrumentation amplifier. The resulting signal contained a relatively small amount of power at the center tone frequency due to bridge balance. Consequently, we could amplify the side tones significantly above system noise at the analog-to-digital converter.

The performance of the circuit was judged in terms of SNR of the center tone GMR sensors which is approximately 100 dB. The center tone was selected for this purpose because it is primarily limited by the circuit components and signal processing whereas SNR of the side tones is highly dependent on the magnetoresistance of sensors. The side tone SNR was approximately 75 dB for the GMR sensors and bench-top system used in chapter 7.

4. Signal processing

After converting the signal to digital data, we could filter away all parts of the electrical signal except the very small portion that we were interested in. We could perform a Fourier Transform at the center tone and side tone frequencies to monitor the amplitudes of each. If we monitored only the side tones, then we could reduce the overall power of the signal to something manageable with respect to the tiny changes induced by the MNPs. In principle, we could say that the side tones contained all the change induced by the MNPs.

The amplitude of the center tone was related to the resistance of the GMR sensor as it was balanced against the other three resistors of the Wheatstone Bridge. Tracking the side tones allowed us to monitor the signal of interest, while monitoring the center tone allowed us to validate the sensor stability in terms of temperature drift and leaky surface protection. For example, if buffer solution leaked into the cracks of the surface protection layer, we

would see at least a partial short-circuit condition within the group of GMR strips defining the sensor. In that case, a sudden jump of center tone would be observed, and that sensor data would be removed from assay analysis.

5. Firmware

The firmware coordinates all activities on the PCB including system setup, sensor calibration, signal generation, signal acquisition, signal processing, and communications with user interface. A high-level recipe of this coordination was provided by the author, and the detailed firmware was written by several other students with Michael Sandstedt as lead developer. The copyrighted firmware is listed in Appendix C.

When a start command is issued from the user interface, the firmware first cycles through each of the sensors to select the most appropriate Wheatstone Bridge balance point with the resistance ladder shown in **Fig 6.2A.** These balance points are stored in local memory and paired to the multiplexer address of each sensor so that each sensor is individually balanced throughout the measurement. This technique helps keep all sensors as close as possible to the balance point of the Wheatstone Bridge such that the circuit response remains nearly linear.

During a measurement cycle, the firmware must first assert a valid multiplexer address such that a given GMR sensor is inserted the Wheatstone Bridge as depicted in **Fig. 6.2A-B**. After this, two sinusoids are generated at appropriate amplitudes (selected by user) and frequencies (f_1 and f_2) to drive both the Wheatstone Bridge and magnetic coil. Finally, the firmware must monitor the values generated by the ADC and process them with Fourier Transform so that the center tone f_1 and side tomes at f_1+/-f_2 can be tracked independently. After the measurement cycle is complete, the detected amplitudes of center tone and side tones are sent via Bluetooth to the user interface on an Android smart phone or tablet.

6. User interface

Again, the job of the author was to provide a high-level specification of how the user interface should look and behave. Several other students were responsible for writing the code. We were able to integrate the GMR biosensing system with the hardware, firmware, software, Bluetooth communications, and mechanical enclosure for the zLab biosensing system as shown in **Fig 6.3A-B**.

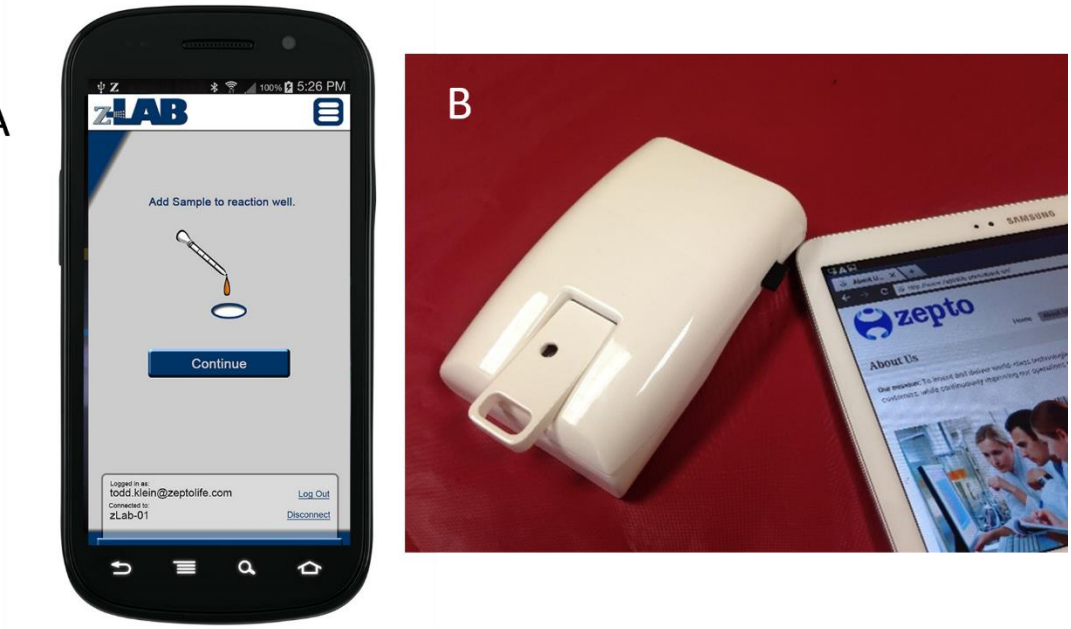

Fig. 6.3. A) Android user interface developed for zLab during Nokia Sensing XCHALLENGE competition. B) zLab prototype.

Since one of the system specifications included low cost, we decided to leverage smart phones to access computing power, network connectivity, and location services. The Android platform was selected due to popularity and easy access to system hardware such as Bluetooth. A BlueSMiRF Bluetooth modem on our PCB was used to wirelessly connect the Android device (smartphone or tablet) to the GMR biosensing system. This allowed the user to start a test and monitor the results.

Screenshots of the user interface are provided in **Fig. 6.4**. [ML2]The interface was designed to guide the user through a few simple steps to set up the assay. Specifically, the user needed to first load a cartridge and turn on the system. At this point, they needed to connect the system to the user interface via Bluetooth, add the sample to a reaction well on the cartridge and add the magnetizer solution to the same reaction well. The system would then begin testing. Finally, the user was presented with data both during and after the assay.

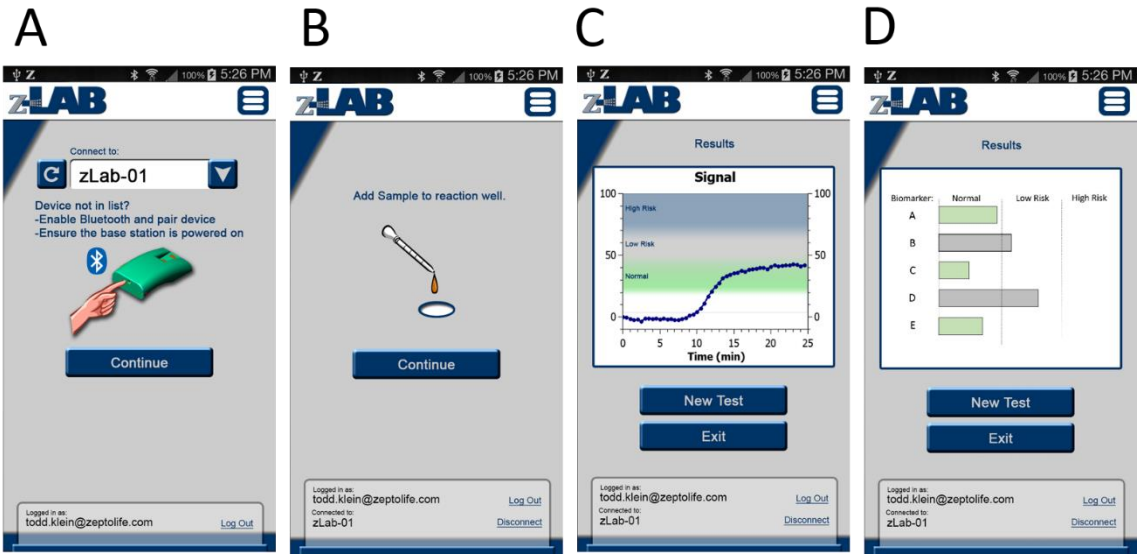

Fig. 6.4. User interface screen shots. A) User is instructed to turn on the base station and connect via Bluetooth. The cartridge has already been inserted. B) User is instructed to add sample to reaction well of cartridge. C) While testing the sample, the user watches the signal develop. D) Multiplex results indicate normal, low-risk, and high-risk ranges for each biomarker.

7. Magnetic field and mechanical system

The author was responsible for simulating and designing the magnetic coil with ferrite core as well as setting the relative position of base station PCB, magnet, and GMR biosensors within the cartridge. A professional designer was responsible for drawing the CAD files for manufacture of the enclosure shown in **Fig. 6.3B**. The most critical aspect of this design was the relative position of the GMR sensors over the top of the magnetic coil with ferrite core.

The coil core simulation results and consequent mechanical requirements are illustrated in **Fig. 6.5A-D**. The simulation was performed in FEMM 4.2 software, and the parameters for simulation were as follows: Coil magnet wire: 400 turns 27AWG magnet wire with 0.2 Amps. Ferrite core: Relative permeability 4400 and coercivity 13 A/m. The width of the coil core gap is 17 mm, and the width of the GMR chip is 16 mm, but all the sensors are contained within 1.2 mm from the chip center. As a safety factor, we require the field within 2 mm of the chip center to be within the acceptable range. From simulation results in Chapter 3, we decided to tolerate a maximum of 1% variation in field intensity. The field intensity along the x-direction is most critical, and its rate of change varies as we adjust GMR chip height along the z-direction. From the results shown in **Fig. 6.5D**, we find an optimal GMR sensor surface height of approximately 3 mm with tolerance of +/- 1.5 mm.

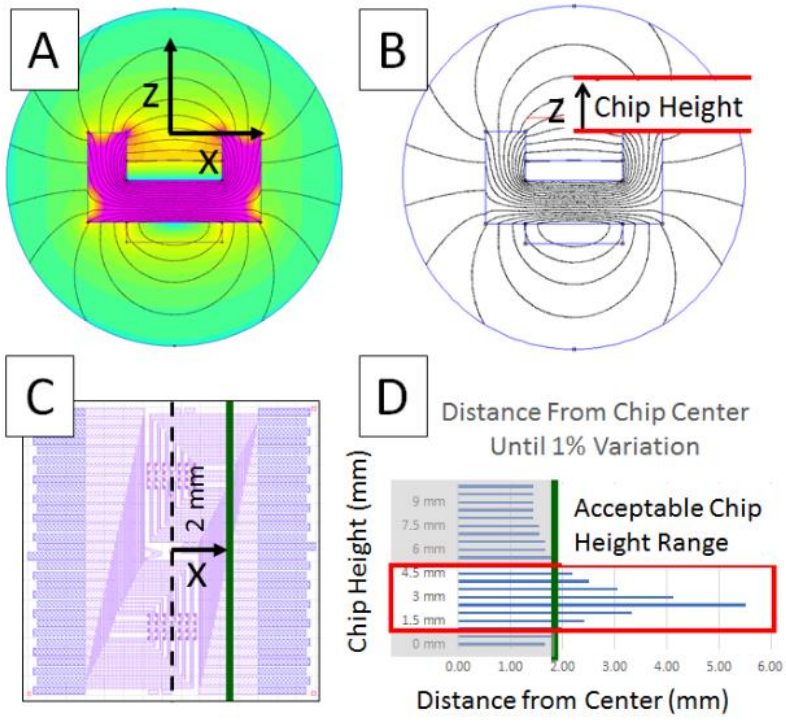

Fig. 6.5. A) FEMM simulation result illustrating field in x-z plane along center of coil core. B) Chip height along z-axis is critical to GMR applied field uniformity along chip surface. C) 2 mm from center of GMR chip includes a safety factor for mechanical tolerance. D) Acceptable height of GMR chip was selected based on distance from center with less than 1% variation in x-component of applied field.

8. Conclusion

The author led the development of PCB, firmware, and software for both a bench-top and a hand-held GMR biosensing system. In addition, he led the overall integration of the hand-held system, which is a system composed of magnetic materials with spintronics properties, a magnetic coil with ferrite core, electronic components assembled on a PCB, a mechanical enclosure, analog and digital signal processing, firmware programming, user interface programming on both a PC and an Android smartphone with communications over both USB and Bluetooth. The net result was the development of both a bench-top and a handheld GMR biosensing system as shown in **Fig. 6.6**. These systems, along with the results on the bench-top system described in Chapter 7, were our main contribution to the field of GMR biosensing.

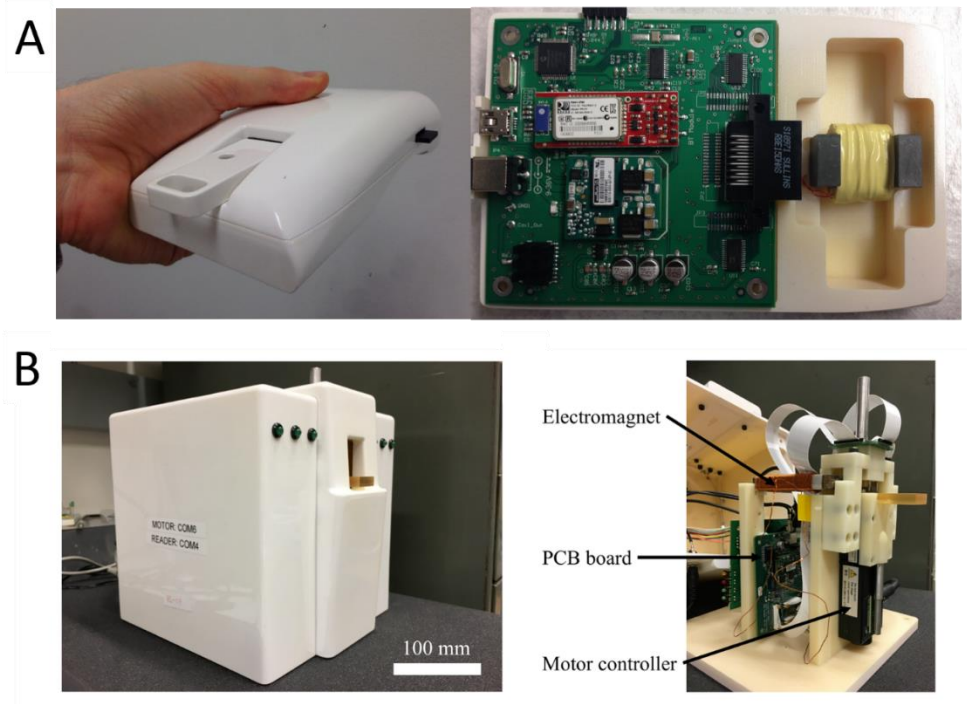

Fig. 6.6. A) Hand-held version of GMR biosensing system. B) Bench-top version of GMR biosensing system.

Chapter 7. Multiplex Detection of Ovarian Cancer Biomarkers

1. Introduction

With the bench-top version of the GMR biosensing system, we aimed to demonstrate a multiplexed ovarian cancer assay. This chapter provides a description of this demonstration using a cocktail of CA-125 II, HE-4, and IL-6 mixed in PBS buffer solution. The mixture of proteins created a rough approximation of a patient blood sample with various concentrations of each analyte for a multiplex measurement. The long-term goal of this research is to develop a low-cost, easy-to-use, highly sensitive test for the detection of ovarian cancer in its early stages when treatments are most effective, as well as for detection of disease recurrence.

To date, no ovarian cancer assay has been produced with adequate sensitivity and specificity to be FDA approved for ovarian cancer screening in the general population. On the other hand, data presented in this chapter demonstrates the potential of our GMR biosensing system to provide a portable, multiplexed, and highly sensitive detection system which can perform such an assay in the future.

2. Multiplex ovarian cancer assay with GMR biosensing

2.1. Surface functionalization and multiplex assay

Similar to the work described in **Appendix B**, we first validated the multiplex assay on glass slides. The results were used to optimize and verify a variety of aspects in the assay including surface chemistry, capture antibody concentrations, detection antibody concentrations, and interference of each component within the multiplex assay. Regarding interference, we first noted the very low background signal which indicated low binding

between surface chemistry and all other assay components. Furthermore, we ran samples spiked with different permutations of the three proteins to confirm that the assays did not interfere. Finally, we included results from a sample of buffer that had no additional proteins so that we could confirm that detection antibodies did not bind directly to capture antibodies of the same assay. As expected, the results indeed indicated low interference, and that BSA is shown to be an effective blocking agent.

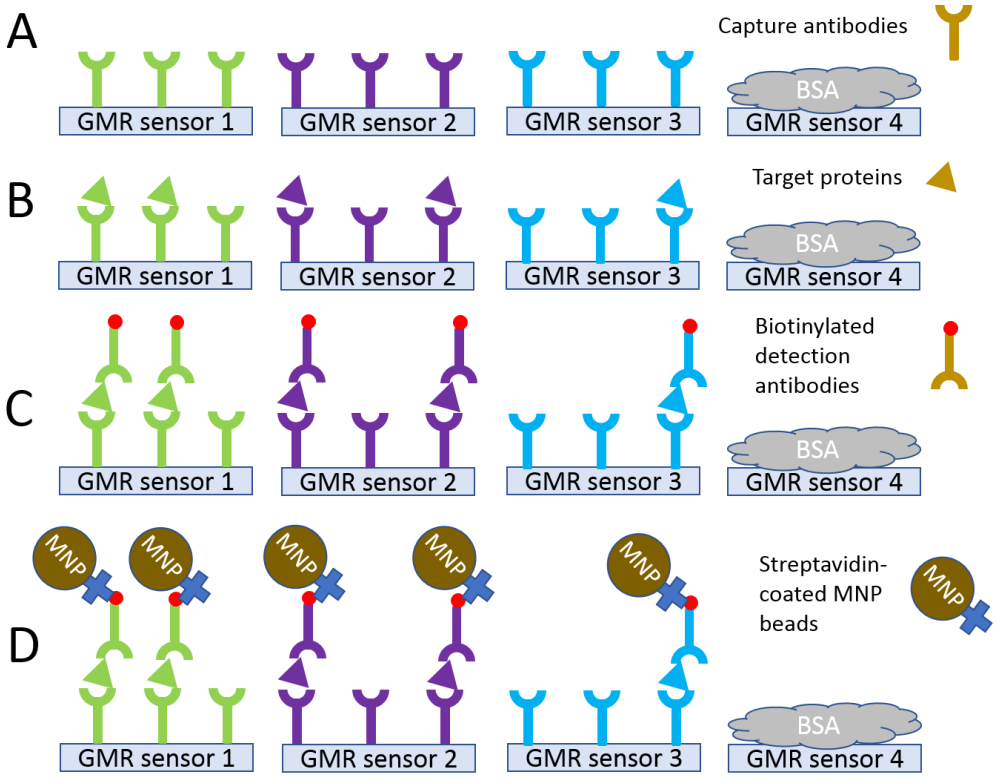

Fig. 7.1. Assay sequence: A) Four groups of four GMR sensors are spotted with either capture antibodies or BSA to define the function of each sensor. B) Sensors are incubated with sample so that each specific target protein (CA-125, HE-4, and IL-6) selectively binds to capture antibodies. C) Sensors are incubated with biotinylated detection antibodies which selectively bind to each specific target protein. D) GMR signals are monitored while incubated with streptavidin-coated MNP beads. [53]

The GMR biosensor assay sequence is illustrated in **Fig. 7.1**. Multiplex GMR biosensor experiments were performed with three different target proteins on a single chip. A

negative control of BSA (10 mg/mL) along with three capture antibodies (CA-125, HE-4, and IL-6, each at 500ug/mL) were robotically spotted onto individual sensors in replicates of four. A total of 16 sensors were used during each multiplex experiment. A spotting volume of approximately 1.2nL was selected to completely cover the sensor. Incubation for 24 hours at a temperature of 4 ºC and approximately 90% relative humidity was performed to immobilize capture antibodies on the GMR surface. After being rinsed with an immunoassay stabilizer (SC01-1000, Surmodics) three times to remove unbound capture antibodies, the chip was washed with PBST (0.02% Tween 20 in PBS) and DI water, then dried with nitrogen gas. A reaction well consisting of a laser-cut acrylic ring was attached to the GMR chip surface with silicone adhesive. The purpose of the reaction well was simply to contain fluids while they incubate over the sensor surface. A blocking buffer containing BSA was added for one hour to block unbound surface sites. The blocking buffer was then removed, and the chip was washed with PBST buffer.

PBS was spiked with mixtures of CA-125, HE-4, and IL-6 at a variety of concentrations to produce dose response curves for each assay. For each concentration along the dose response curves, we added 100uL of the PBS-protein solution and incubated for 60 minutes at room temperature. After this multiplex incubation, the sensors were washed with PBST buffer. The three biotinylated detection antibodies for CA-125, HE-4, and IL-6, each at a concentration of 5ug/mL in PBS, were then added to the reaction well and incubated at room temperature for 60 minutes. Finally, the chip was washed with PBST and dried with nitrogen gas.

2.2. Signal Collection

The GMR chip was loaded into the benchtop GMR base station and 30uL PBS was added to the reaction well. After 10 minutes of electronics warmup time, the data collection

began. We allowed the signal to generate a baseline for 3 minutes before adding the MNPs. The magnetic labels used in this experiment were Ademtec 200nm beads. Each bead was composed of approximately 1000 MNPs with magnetic moments of 2.3 x 10^{-16} emu, held together by a polymer matrix. The VSM data and analysis of the Ademtec beads can be found in **Fig. 2.5B**. These beads were selected for high magnetic volume and therefore high moment per bead binding event. One drawback to using such large beads was the potential for non-specific signal due to beads settling out of solution. For this reason, we always included a negative control group covered in BSA so that any background signal can be subtracted.

3. Results and discussion

The multiplex ovarian cancer assay signals presented in this work (**Fig. 7.2**) were measured with the bench-top GMR base station in **Fig. 6.6B**. After converting the center tone and side tone voltages to magnetoresistance amplitude, we were left with noise plus drift of +/-5ppm relative to the sensor resistance. After adding MNPs to the reaction well, the signal clearly rose to levels as high as 2,000 ppm in proportion to protein concentration as illustrated in **Fig. 7.2D**.

Data points on each dose-response curve represent 4 replicate GMR sensor signals on a chip. The various concentrations required to build each dose-response curve were attained by mixing all three target proteins at each specified concentration for each data point. The mixtures of target proteins were each incubated on separate GMR chips. The dynamic range demonstrated for each protein concentration was two orders of magnitude. This was adequate for the detection of biologically relevant concentrations of the biomarkers, but somewhat limited on the upper end of concentration. On the low end, the limit of detection

was less than 8pg/mL for both HE-4 and IL-6 and less than 4 units/mL for CA-125. These limits of detection are acceptable for medically relevant assays [6].

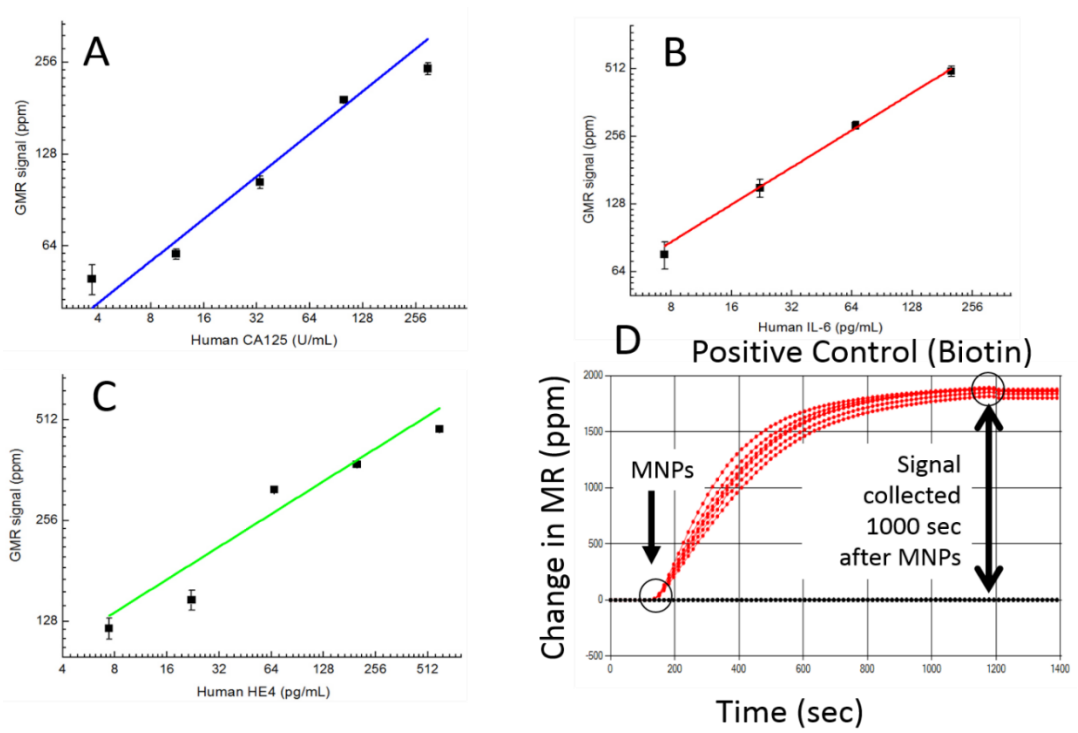

Fig. 7.2. Data points on each dose-response curve represent 4 replicate GMR sensor signals. A) Dose-response curve for human CA-125 in multiplex assay. B) Dose-response curve for human IL-6 in multiplex assay. C) Dose-response curve for human HE-4 in multiplex assay. D) Positive control signal from biotin-BSA. [53]

Chapter 8. Conclusion and Outlook

In this chapter, we will summarize technological achievements, discuss the GMR biosensor designs presented in previous chapters, and provide possible directions for further research.

1. Technological achievements

We developed desk-top and hand-held versions of a GMR biosensing system that could contribute to the early detection of ovarian cancer and serve as platforms for detecting a wide variety of other biomarkers. The prototype system that can deliver the required performance in terms of high sensitivity multiplex assays in a portable format with enough flexibility to serve as a platform for ovarian cancer and many other diseases. This achievement was made possible by a multidisciplinary team of graduate students, undergraduate students, professors, and professional engineers in the areas of medicine and pathology, biochemistry, electrical engineering, mechanical engineering, firmware and software development. Team members are listed in Appendix A. The magnitude of this achievement is reflected in our success with the hand-held system at the Nokia Sensing XCHALLENGE competition as a Distinguished Award recipient [90].

We achieved a multiplex detection of CA-125, HE-4, and IL-6 in PBS buffer with a limit of detection below 10pg/mL. The system was constructed from the linear GMR biosensor array whose magnetoresistance signals were monitored by a nearly balanced (per each sensor) Wheatstone Bridge circuit. In the hand-held version of our system, the sensor array contained 29 sensors to take advantage of the 30-pin connector for smart phones at the time. The benchtop system used an array of 64 sensors. Each sensor can be printed with the desired capture antibodies to serve as selective binding sites for the biomarkers within

the multiplex assay, so many biomarkers can, in theory, be tested simultaneously. In practice, the total number of biomarkers required for a multiplex assay is expected to be at least three, and maybe more than five. Exceeding this number of biomarkers would be desirable in a screening application, but a lot of work will need to be done on the biological side to deliver the required sensitivity and specificity.

We believe the GMR biosensor can contribute to ovarian cancer screening if assays with appropriate sensitivity and specificity are developed. We have yet to see if the system can be affordable, fully automated, and able of competing with central laboratories in terms of precision and reliability, but development in those areas is already underway.

2. Analysis of GMR biosensor models

2.1. Linear sensor

2.1.1. Advantages and current limitations to the linear sensor

As noted in chapter 3, the main advantage of the linear sensor for GMR biosensing applications is that it is based on a quantitative theoretical model that is already well-established in the literature.

Also noted in chapter 3 were current limitations to the linear sensor: 1) It does not naturally cover the desired amount of space in the biosensor as defined by the capture antibody printing spots, 2) MNP signals vary greatly according to location relative to sensor strips, and 3) Sensitivity is limited by the necessary large shape anisotropy and a 90-degree field orientation as depicted in **Fig. 3.2B**.

2.1.2. Future directions for linear sensor

One way to combat the signal variation due to MNP location is to pattern capture antibodies so that they align more uniformly with the sensor strips. This way, specific binding on the assay would occur only in the predefined sensor locations to maximize

signal and reduce variation. This would have a tremendous impact on the linear sensor performance, but the requirements for alignment in the sub-μm range are not trivial.

It is also possible that the sensitivity of linear sensors will be even further improved with the use of high-moment CoFe MNPs. It is important, then, to develop more reliable ways to consistently produce these MNPs in large quantities.

2.2. Large area sensor with multidomain switching

2.2.1. Advantages and limitations of the large area sensor

As noted in chapter 4, the large area and low aspect-ratio sensor directly addresses sensitivity limitations by orienting the applied field nearly parallel to the easy axis defined by induced anisotropy during film deposition. In addition, it is simple to fabricate and does not have any gaps in the sensing area.

Also noted in chapter 4 were these limitations of the large area sensor: 1) It requires complex signal processing, 2) The model is not as rigorously quantitative as the linear sensor model, so signal-concentration correlations can only be determined experimentally, and 3) We were unable to reproduce the desired multidomain switching characteristic observed in past successful implementations.

2.2.2. Future directions for large area sensor with multidomain switching

To further develop the large area and low aspect-ratio sensor, we should devise a convenient method for signal collection. This does not necessarily need to be a complicated procedure, but the electronics still need to be designed. In addition, we may wish to continue work on the multidomain switching array to build a more quantifiable theoretical model as well as a system whose functionality depends more heavily on patterning and less on the exact composition of GMR films. The multidomain switching array lends itself more easily to micromagnetic simulation, so we may expect a more tractable sensor.

Finally, we need to ramp up production of high-moment CoFe MNPs. The value of a small-diameter, high-moment MNP should not be underestimated, but neither should the difficulty of this technical challenge. To date, there is no commercial source of CoFe MNPs. After gaining access to CoFe MNPs, we will want to compare the multidomain switching array to our previous large area sensor with multidomain switching to see if it can be used to enhance the large area sensor capabilities.

2.3. Domain wall sensor

2.3.1. Advantages and limitations to the domain wall sensor

The domain wall sensor has the capability of detecting several MNPs. With such high sensitivity, this type of sensor has the potential to act as a digital count of magnetic beads. On the other hand, it is hard to see a clear pathway for translating this precision to ordinary biosensing, which naturally involves a large number of binding sites randomly distributed across the sensing surface. On-chip CMOS electronics could theoretically solve this problem, but developing such technology was outside the scope of our work.

2.3.2. Future directions for the domain wall sensor

The most straightforward approach to increasing the sensing area would be to integrate the domain wall sensor with an CMOS array. In this way, each domain wall sensor would be individually addressable, but we could include thousands of sensors on a single sensing chip.

Prior to CMOS integration, it is advised to make sure there is a real sensitivity advantage over a linear sensor of the same size. To make this comparison, it is recommended that the same structure be used in both ways. Although the curved structure is not a standard linear sensor design, the high aspect ratio makes it similar and possibly

close enough to be useful. By simply adding or removing the head-to-head domain wall, we can easily switch between the two structures.

The concept of adding or removing a domain wall dynamically during testing carries the extra benefit of reducing the chance of non-specific binding. For example, we can imagine a sensing scheme where the domain wall is added to the system only for short periods of time to check the strength of the de-pinning field. In between such measurements, we simply remove the domain wall and thereby avoid the potential for non-specific binding during that time.

3. Future work

3.1. Ovarian cancer assay

The GMR biosensor system has demonstrated a multiplexed assay with a mixture of CA-125, HE-4, and IL-6 in the sample. The limits of detection for all three proteins are acceptable for real biological samples, and they can be run on a portable system. Furthermore, system automation and microfluidic integration is currently in progress for the hand-held magnetic biosensor. The next step that should be taken is to develop GMR assays for LRG and Nectin-4 [91, 92]. After that, we will be ready to validate the 5-biomarker multiplexed assay in serum samples.

3.2. Sensor design research

For now, the magnetic biosensing platform is built around a linear sensor design, with biologically relevant sensitivity and dynamic range. Aided by recent research [63], we are excited by the new theoretical model and note the importance of high-moment CoFe MNPs in the use of large area sensors. It is advisable that further work on assay development with CoFe MNPs and large area sensors is performed to enhance sensing capabilities. Domain wall sensors provide an exciting pathway toward single molecule detection, but the

practical challenge of sensing many MNPs spread across a surface will require creative design and perhaps more advanced spintronic materials.

Author Publications and Presentations

PUBLICATIONS

- **Todd Klein**, Wei Wang, Lina Yu, Kai Wu, Kristin Boylan, Rachel Isakkson Vogel, Amy Skubitz, and Jian-Ping Wang. **(To Be Submitted)** "Multiplex detection of ovarian cancer biomarkers with GMR biosensor array."

- Kai Wu, **Todd Klein**, Venkatramana D. Krishna, Diqing Su, Andres M. Perez, and Jian-Ping Wang. "Portable GMR handheld platform for the detection of Influenza A virus," *ACS sensors*, vol. 2, no. 11, pp. 1594-1601, 2017.

- Yinglong Feng, Jinming Liu, **Todd Klein**, Kai Wu and JianPing Wang, "Localized detection of reversal nucleation generated by high moment magnetic nanoparticles using a large-area magnetic sensor," *Journal of Applied Physics*, vol. 122, no. 12, p. 123901. (2017)

- Yi Wang, Wei Wang, Lina Yu, Liang Tu, Yinglong Feng, **Todd Klein** and JianPing Wang, "Giant magnetoresistive-based biosensing probe station system for multiplex protein assays," *Biosensors and Bioelectronics*, vol. 70: pp. 61-68. (2015)

- **Todd Klein**, Yi Wang, Liang Tu, Lina Yu, Yinglong Feng, Wei Wang, and Jian-Ping Wang. "Comparative analysis of several GMR strip sensor configurations for biological applications," *Sensors and Actuators A: Physical* vol. 216: pp. 349-354 (2014)

- **Todd Klein**, Jonathan Lee, Wei Wang, Tofizur Rahman, Rachel Isaksson-Vogel, and Jian-Ping Wang, "Interaction of domain walls and magnetic nanoparticles in giant magnetoresistive nanostripes for biological applications," *IEEE Transactions on Magnetics* vol. 49, no. 7 (2013)

- **Todd Klein**, Daniel Dorroh, Yuanpeng Li, and Jian-Ping Wang, "Quantitative analysis of interaction between domain walls and magnetic nanoparticles," *Journal of Applied Physics* vol. 109, p. 07D506 (2011)

- Andrew Lyle, Jonathan Harms, **Todd Klein**, August Lentsch, Daniel Martens, Angeline Klemm, and Jian-Ping Wang, "Spin transfer torque programming dipole coupled nanomagnet arrays," *Appl. Phys. Letters*. Vol. 100, p. 012402 (2012)

- Andrew Lyle, Jonathan Harms, **Todd Klein**, August Lentsch, Angeline Klemm, Daniel Martens, and Jian-Ping Wang, "Integration of spintronic interface for nanomagnetic arrays," *AIP Advances* vol. 1, p. 042177 (2011)

- Liang Tu, **Todd Klein**, Wei Wang, Yinglong, Feng, and Jian-Ping Wang, "Measurement of Brownian and Neel relaxation of magnetic nanoparticles by a mixing-frequency method," *IEEE Transactions on Magnetics* vol. 49, no. 1 (2013)

- Wei Wang, Yi Wang, Liang Tu, **Todd Klein**, Yinglong Feng, and Jian-Ping Wang, "Surface modification for protein and DNA immobilization onto GMR biosensor," *IEEE Transactions on Magnetics* – Magnetic Carriers Conference 2012, p. 161 (2012)

- Liang Tu, Yinglong Feng, **Todd Klein**, Wei Wang, and Jian-Ping Wang, "Measurement of Brownian of magnetic nanoparticles by a multitone mixing-frequency method," *IEEE Transactions on Magnetics* vol. 48, no. 11, pp. 3513-3516 (2012)

CONFERENCE PRESENTATIONS

- Poster DQ-15: 58th Annual Conference on Magnetism and Magnetic Materials: Denver, Colorado
  "Domain nucleation array and its interaction with magnetic nanoparticles,"
  Todd Klein, Yinglong Feng, Lina Yu, Yi Wang, and Jian-Ping Wang
  (November 6, 2013)

- Institute for Engineering in Medicine Annual Conference and Retreat: Minneapolis, Minnesota
  "A smartphone-integrated handheld GMR system for disease early detection and monitoring,"
  Todd Klein, Liang Tu, Yi wang, Yinglong Feng, Hoang Tran, Xiqian Zheng, Michael Sandstedt, Jonathon Pechuman, Samiha Sultana, Rachel Isaksson-Vogel, Kristin Boylan, Amy Skubitz, and Jian-Ping Wang (September 23, 2013)

- Poster EU-06: 12th Joint MMM/Intermag Conference: Chicago, Illinois
  "Interaction of domain walls and magnetic nanoparticles in giant magnetoresistive nanostripes for biological applications," **Todd Klein**, Jonathan Lee, Wei Wang, Tofizur Rahman, Rachel Isaksson-Vogel, and Jian-Ping Wang
  (January 17, 2013)

- Poster GP-12: 55th Annual Conference on Magnetism and Magnetic Materials: Atlanta, Georgia
  "Quantitative analysis of interaction between domain walls and magnetic nanoparticles"
  Todd Klein, Daniel Dorroh, Yuanpeng Li, and Jian-Ping Wang
  (November 18, 2010)

Bibliography

[1] R. Siegel, K. Miller and A. Jemal, "Cancer statistics, 2016," *CA: a cancer journal for clinicians,* vol. 66, no. 1, pp. 7-30, 2016.

[2] G. Jayson, E. Kohn, H. Kitchener and J. Ledermann, "Ovarian cancer," *The Lancet,* vol. 384, no. 9951, pp. 1376-1388, 2014.

[3] "Surveillance, Epidemiology, and End Results (SEER) Program (www.seer.cancer.gov) Research Data (1973-2014)," National Cancer Institute, DCCPS, Surveillance Research Program, April 2017, based on the November 2016 submission.

[4] R. Bast Jr, T. Klug, E. John, E. Jenison, J. Niloff, H. Lazarus, R. Berkowitz, T. Leavitt, C. Griffiths, L. Parker and V. Zurawski Jr, "A radioimmunoassay using a monoclonal antibody to monitor the course of epithelial ovarian cancer," *New England journal of medicine,* vol. 309, no. 15, pp. 883-887, 1983.

[5] I. Jacobs and R. Bast Jr, "The CA 125 tumour-associated antigen: a review of the literature," *Human reproduction,* vol. 4, no. 1, pp. 1-2, 1989.

[6] R. Moore, D. McMeekin, A. Brown, P. DiSilvestro, M. Miller, W. Allard, W. Gajewski, R. Kurman, R. Bast and S. Skates, " A novel multiple marker bioassay utilizing HE4 and CA125 for the prediction of ovarian cancer in patients with a pelvic mass," *Gynecologic oncology,* vol. 112, no. 1, pp. 40-46, 2009.

[7] F. Ueland, "A perspective on ovarian cancer biomarkers: past, present and yet-to-come," *Diagnostics,* vol. 7, no. 1, p. 14, 2017.

[8] F. Ueland, C. Desimone, L. Seamon, R. Miller, S. Goodrich, I. Podzielinski, L. Sokoll, A. Smith, J. van Nagell Jr and Z. Zhang, "Effectiveness of a multivariate index assay in the preoperative assessment of ovarian tumors," *Obstetrics & Gynecology,* vol. 117, no. 6, pp. 1289-1297, 2011.

[9] R. Coleman, T. Herzog, D. Chan, D. Munroe, T. Pappas, A. Smith, Z. Zhang and J. Wolf, "Validation of a second-generation multivariate index assay for malignancy risk of adnexal masses," *American journal of obstetrics and gynecology,* vol. 215, no. 1, pp. 82-e1, 2016.

[10] F. Li, R. Tie, K. Chang, F. Wang, S. Deng, W. Lu, L. Yu and M. Chen, "Does risk for ovarian malignancy algorithm excel human epididymis protein 4 and CA125 in predicting epithelial ovarian cancer: A meta-analysis," *BMC Cancer,* vol. 12, no. 258, 2012.

[11] K. Buechler, S. Moi, B. Noar and others, "Simultaneous detection of seven drugs of abuse by the Triage(tm) panel for drugs of abuse," *Clinical Chemistry,* vol. 38, no. 9, pp. 1678-1684, 1992.

[12] J. Melin, G. Rundström, C. Peterson and others, "A multiplexed point-of-care assay for C-reactive protein and N-terminal pro-brain natriuretic peptide," *Analytical Biochemistry,* vol. 409, no. 1, pp. 7-13, 2011.

[13] P. Petrou, S. Kakabakos and K. Misiakos, "Silicon optocouplers for biosensing," *International Journal of Nanotechnology,* vol. 6, no. 1-2, pp. 4-17, 2008.

[14] Y. Zhou, H. Yin, X. Li, Z. Li, S. Ai and H. Lin, "Electrochemical biosensor for protein kinase A activity assay based on gold nanoparticles-carbon nanospheres, phos-tag-biotin and β-galactosidase," *Biosensors and Bioelectronics,* vol. 15, no. 86, pp. 508-515, 2016.

[15] R. Gupta, R. Pandya, T. Sieffert, M. Meyyappan and J. Koehne, "Multiplexed electrochemical immunosensor for label-free detection of cardiac markers using a carbon nanofiber array chip," *Journal of Electroanalytical Chemistry,* vol. 15, no. 773, pp. 53-62, 2016.

[16] G. Zheng, F. Patolsky, Y. Cui, W. Wang and C. Lieber, "Multiplexed electrical detection of cancer markers with nanowire sensor arrays," *Nature biotechnology,* vol. 23, no. 10, pp. 1294-1301, 2005.

[17] N. Gao, T. Gao, X. Yang and others, "Specific detection of biomolecules in physiological solutions using graphene transistor biosensors," *Proceedings of the National Academy of Sciences,* vol. 113, no. 51, pp. 14633-14638, 2016.

[18] M. Ramanathan, M. Patil, R. Epur and others, "Gold-coated carbon nanotube electrode arrays: Immunosensors for impedimetric detection of bone biomarkers," *Biosensors and Bioelectronics,* vol. 77, pp. 580-588, 2016.

[19] Y. Li, B. Srinivasan, Y. Jing, X. Yoa, M. Hugger, J. Wang and C. Xing, "Nanomagnetic competition assay for low-adundance protein biomarker quantification in unprocessed human sera," *Journal of American Chemical Society,* no. 132, pp. 4388-4392, 2010.

[20] R. Gaster, D. Hall, C. Nielsen, S. Osterfeld, H. Yu, K. Mach, R. Wilson, B. Murmann, J. Liao, S. Gambhir and S. Wang, "Matrix-insensitive protein assays push the limits of biosensors in medicine," *Nature medicine,* vol. 15, no. 11, pp. 1327-1332, 2009.

[21] Y. Wang, W. Wang, L. Yu, L. Tu, Y. Feng, T. Klein and W. JP, "Giant magnetoresistive-based biosensing probe station system for multiplex protein assays," *Biosensors and Bioelectronics,* vol. 70, pp. 61-68, 2015.

[22] J. Wang, Z. Yao, T. Lei and A. Poon, "Silicon coupled-resonator-optical-waveguide-basd biosensors using light-scattering pattern recognition with pixelized mode-field-intensity distributions," *Scientific Reports,* vol. 4, no. 7528, pp. 1-9, 2014.

[23] M. Clinic, "Human Epididymis Protein 4, Serum," [Online]. Available: https://www.mayomedicallaboratories.com/test-catalog/Performance/62137.

[24] W. Shen, M. Schrag and G. Xiao, "Quantitative detection of DNA labeled with magnetic nanoparticles using arrays of MgO-based magnetic tunnel junction sensors," *Appl. Phys. Lett,* vol. 93, p. 033903, 2008.

[25] L. Hu, H. Yu, S.-J. Han, S. Osterfeld, R. White, N. Pourmand and others, "Giant mangetoresistive sensors for DNA microarray," *IEEE Trans. Magn.,* vol. 44, pp. 3983-3991, 2008.

[26] H. Brucki and M. Panhorst, "Magnetic particles as markers and carriers of biomolecules," *IEE Proc.-Nanobiotechnology,* vol. 152, pp. 41-46, 2005.

[27] R. Chaves, D. Bensimon and P. Freitas, "Single molecule actuation and detection on a lab-on-a-chip magnetoresistive platform," *Journal of Applied Physics,* vol. 109, no. 6, pp. 064702-064702, 2011.

[28] S. Mulvaney, K. Myers, P. Sheehan and L. Whitman, "Attomolar protein detection in complex sample matrices with semi-homogenous fluidic force discrimination assays," *Bionsens. Bioelectron.,* vol. 24, pp. 1109-1115, 2009.

[29] K. Kim and J. Park, "Magnetic force-based multiplexed immunoassay using superparamagnetic nanoparticles in a microfluidic channel," *Lab Chip,* vol. 5, pp. 657-664, 2005.

[30] R. Gaster, L. Xu, S.-J. Han, R. Wilson, D. Hall, S. Osterfeld and others, "Quantification of protein interactions and solution transport using high-density GMR sensor arrays," *Nat. Nanotechnol.,* vol. 6, pp. 314-320, 2011.

[31] V. Morozov, S. Groves, M. Turell and C. Bailey, "Three minutes-long electrophoretically assisted zeptomolar microfluidic imuunoassay with magnetic-beads detection," *J. Am. Chem. Soc.,* vol. 129, pp. 12628-12629, 2007.

[32] T. Aytur, J. Foley, M. Anwar, B. Boser, E. Harris and P. Beatty, "A novel magnetic bead bioassay platform using a microchip-basd sensor for infectious disease diagnosis," *J. Immunol. Methods,* vol. 314, pp. 21-29, 2006.

[33] B. Srinivassan, Y. Li, Y. Jing, Y. Xu, X. Yao, C. Xing and J. Wang, "A detection system based on giant magnetoresistive sensors and high-moment magnetic nanoparticles demonstrates zeptomole sensitivity: Potential for personalized medicine," *Angewandte Chemie,* vol. 48, no. 15, pp. 2764-2767, 2009.

[34] B. Srinsivan, Y. Li, Y. Jing and C. Xing, "A three-layer competition-based giant magnetoresistive assay for direct quantification of endoglin from human urine," *Anal. Chem.,* vol. 83, pp. 2996-3002, 2011.

[35] J. Rife, M. Miller, P. Sheehan, C. Tamanaha, M. Tondra and L. Whitman, "Design and performance of GMR sensors for the detection of magnetic microbeads in biosensors," *Sensors and Actuators A: Physical,* vol. 107, pp. 209-218, 2003.

[36] D. Baselt, G. Lee, M. Natesan, S. Metzger, P. Sheehan and R. Colton, "A biosensor based on magnetoresistance technology," *Biosens. Bioelectron.,* vol. 13, pp. 731-739, 1998.

[37] M. Tondra, M. Porter and R. Lipert, "Model for detection of immobilized superparamagnetic nanosphere assay labels using giant magnetoresistive sensors," *J. Vac. Sci. Technol. A,* vol. 18, no. 4, pp. 1125-1129, 2000.

[38] J. Schotter, P. Kamp, A. Becker, A. Puhler, D. Brinkmann, W. Schepper and others, "A biochip based on magnetoresistive sensors," *IEEE Trans. Magn.,* vol. 38, pp. 3365-3367, 2002.

[39] M. Miller, P. Sheehan, R. Edelstein, C. Tamanaha, L. Zhong, S. Bounnak and others, "A DNA array sensor utilizing magnetic microbeads and magnetoelectronic detection," *J. Magn. Mater.,* vol. 225, pp. 138-144, 2001.

[40] P. Haddix, T. Werner, N. Chang and S. Galewsky, "News and views," *Biol. Teach.,* vol. 3, pp. 5-7, 2000.

[41] G. Li, S. Sun, R. Wilson, N. Pourmand and S. Wang, "Spin valve sensors for ultrasensitive detection of superparamagnetic nanoparticles for biological applications," *Sens. Actuators A Phys.,* vol. 126, pp. 98-106, 2006.

[42] C. Tamahana and S. Mulvaney, "Magnetic labeling, detection, and system integration," *Bionsens. Bioelectron.,* vol. 24, pp. 1-13, 2008.

[43] P. Freitas, F. Cardoso, V. Martins, S. Martins, J. Loureiro, J. Amaral and others, "Spintronic platforms for biomedical applications," *Lab Chip,* vol. 12, pp. 546-557, 2012.

[44] J. Gordon and G. Michel, "Discerning trends in multiplex immunoassay technology with potential for resource-limited settings," *Clin. Chem.,* vol. 58, pp. 690-698, 2012.

[45] A. Fert, "Origin, development, and future of spintronics (Nobel Lecture)," *Angew. Chemie Int. Ed.,* vol. 47, pp. 5956-5967, 2008.

[46] E. Fullerton and I. Schuller, "The Nobel Prize in Physics: magnetism and transport at the nanoscale," *ACS Nano.,* vol. 1, pp. 384-389, 2007.

[47] G. Prinz, "Magnetoelectronics," *Science,* vol. 282, pp. 1660-1663, 1998.

[48] V. Martins, J. Germano, F. Cardoso, J. Loureiro, L. Sousa and others, "Challenges and trands in the development of a magnetoresistive biochip portable platform," *J. Magn. Magn. Mater.,* vol. 322, pp. 1665-1663, 2010.

[49] B. De Boer, J. Kahlman, T. Jansen, H. Duric and J. Veen, "An integrated and sensitive detection platform for magneto-resistive biosensors," *Biosensors and Bioelectronics,* vol. 22, no. 9, pp. 2366-2370, 2007.

[50] J. Germano, V. Martins, F. a Cardoso, T. Almeida, L. Sousa, P. Freitas and others, "A portable and autonomoous magnetic detection platform for biosensing," *Sensors,* vol. 9, pp. 4119-4137, 2009.

[51] D. Hall and R. Gaster, "A 256 channel magnetoresistive biosensor microarray for quantitative proteomics," *IEEE 2011 Symp. on VSLIC,* pp. 174-175, 2011.

[52] K. Wu, T. Klein, V. Krishna, D. Su, A. Perez and J. Wang, "Portable GMR handheld platform for the detection of Influenza A virus," *ACS sensors,* vol. 2, no. 11, pp. 1594-1601, 2017.

[53] T. Klein, W. Wang, L. Yu, K. Wu, K. Boylan, R. Isakkson Vogel, A. Skubitz and J. Wang, "Multiplex detection of ovarian cancer biomarkers with a GMR biosensor array," *(To be submitted).*

[54] G. Li, S. Wang and S. Sun, "Model and experiment of detecting multiple magnetic nanoparticles as biomolecular labels by spin valve sensors.," *Magnetics, IEEE Transactions on,* vol. 40, no. 4, pp. 3000-3002, 2004.

[55] T. Klein, Y. Wang, L. Tu, L. Yu, Y. Feng, W. Wang and J. Wang, "Comparative analysis of several GMR strip sensor configurations for biological applications," *Sensors and Actuators A: Physical,* vol. 216, pp. 349-354, 2014.

[56] G. Li, S. Sun and S. Wang, "Spin valve biosensors: Signal dependence on nanoparticle position," *J. Appl. Phys.,* vol. 99, p. 08P107, 2006.

[57] T. Klein, D. Dorroh, Y. Li and J. Wang, "Quantitative analysis of interaction between domain walls and magnetic nanoparticles," *Journal of Applied Physics,* vol. 109, no. 7, p. 07D506, 2011.

[58] T. Klein, J. Lee, W. Wang, T. Rahman, R. Isaksson Vogel and J. Wang, "Interaction of domain walls and magnetic nanoparticles in giant magnetoresistive nanostrips for biological applications," *IEEE Tans. Mag. Conference Proceedings,* 2013.

[59] R. O'Handley, Modern magnetic materials: principles and applications, 2000.

[60] H. N. Bertram, "Two-dimensional fields," in *Theory of Magnetic Recording*, Cambridge, Press Syndicate of the University of Cambridge, 1994, pp. 28-31.

[61] N. Mott, "The electrical conductivity of transition metals," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences,* vol. 153, no. 880, pp. 699-717, 1936.

[62] S. Thompson, "The discovery, development and future of GMR: The Nobel Prize 2007," *Journal of Physics D: Applied Physics,* vol. 41, no. 9, p. 093001, 2008.

[63] Y. Feng, J. Liu, T. Klein, K. Wu and J. Wang, "Localized detection of reversal nucleation generated by high moment magnetic nanoparticles using a large-area magnetic sensor," *Journal of Applied Physics,* vol. 122, no. 12, p. 123901, 2017.

[64] R. Koch, J. Deak, D. Abraham, P. Trouilloud, R. Altman, Y. Lu, W. Gallagher, R. Scheuerlein, K. Roche and S. Parkin, "Magnetization reversal in micron-sized magnetic thin films," *Physical review letters,* vol. 81, no. 20, p. 4512, 1998.

[65] W. Brown Jr, "Rigorous calculation of the nucleation field in a ferromagnetic film or plate," *Physical Review,* vol. 124, no. 5, p. 1348, 1961.

[66] Y. Jing, S. He, T. Kline, Y. Xu and J. Wang, "High-magnetic-moment nanoparticles for biomedicine," *Engineering in Medicine and Biology Society,* vol. Annual International Conference of the IEEE 2009, pp. 4483-4486, 2009.

[67] H. Ma, J. Zhou, M. Yang, Y. Liu, S. Zeng, W. Zhou, L. Zhang, T. Venkatesan and Y. Feng, "Giant crystalline anisotropic magnetoresistance in nonmagnetic perovskite oxide heterostructures," *Physical Review B,* vol. 95, no. 15, p. 155314, 2017.

[68] S. Mao, Y. Chen, F. Liu, X. Chen, B. Xu, P. Lu, M. Patwari, H. Xi, C. Chang, B. Miller and D. Menard, "Commercial TMR heads for hard disk drives: characterization and extendibility at 300 gbit/in/sup 2," *IEEE transactions on magnetics,* vol. 42, no. 2, pp. 97-102, 2006.

[69] M. Biotech, "MACS MicroBeads," [Online]. Available: http://www.miltenyibiotec.com/en/products-and-services/macs-cell-separation/macs-technology/microbeads_dp.aspx.

[70] D. Heim, R. Fontana, C. Tsang, V. Speriosu, B. Gurney and M. Williams, "Design and operation of spin valve sensors," *IEEE Transactions on Magnetics,* vol. 30, no. 2, pp. 316-321, 1994.

[71] J. Daughton, J. Brown, E. Chen, R. Beech, A. Pohm and W. Kude, "Magnetic field sensors using GMR multilayer," *IEEE Transactions on magnetics,* vol. 30, no. 6, pp. 4608-4610, 1994.

[72] J. Rife, M. Miller, P. Sheehan, C. Tamanaha, M. Tondra and L. Whitman, "Design and performance of GMR sensors for the detection of magnetic microbeads in biosensors," *Sensors and Actuators A: Physical,* vol. 107, pp. 209-218, 2003.

[73] G. Li and S. Wang, "Analytical and micromagnetic modeling for detection of a single magnetic microbead or nanobead by spin valve sensors," *Magn. IEEE Trans,* vol. 39, pp. 3313-3315, 2003.

[74] J. Nordling, R. Millen, H. Bullen and e. al, "Giant magnetoresistance sensors," *Anal. Chem.,* vol. 80, pp. 7930-7939, 2008.

[75] C. Damsgaard and M. Hansen, "Theoretical study of in-plane response of magnetic field sensor to magnetic beads in an in-plane homogenous field," *J. Appl. Phys.,* vol. 103, p. 064512, 2008.

[76] H. A. Ferreira, N. Feliciano, D. L. Graham and P. P. Freitas, "Effect of spin-valve sensor magnetostatic fields on nanobead detection for biochip applications," *Journal of Applied Physics,* no. 97, p. 10Q904, 2005.

[77] C. Little, J. Pellegrino and S. Russek, "Microfluidic platform for mangetic nanoparticle trapping and detection," *IEEE Trans. Magn.,* vol. 49, pp. 3402-3405, 2013.

[78] R. Cowburn, A. Adeyeye and J. Bland, "Magnetic domain formation in lithographically defined antidot Permalloy arrays," *Applied physics letters,* vol. 70, no. 17, pp. 2309-2311, 1997.

[79] M. Donahue and D. Porter, "OOMMF user's guide, version 1.0," 1999.

[80] W. Wang, C. Mu, B. Zhang, Q. Liu and J. Wang, "2DPBC module for OOMMF," 2009.

[81] P. Vavassori, V. Metlushko, B. Ilic, M. Gobbi, M. Donolato, M. Cantoni and R. Bertacco, "Domain wall displacement in Py square ring for single nanometric magnetic bead detection," *Applied Physics Letters,* vol. 93, pp. 203502-1 to 3, 2008.

[82] S. Wang and G. Li, "Advances in giant magnetoresistance biosensors with magnetic nanoparticle tags: Review and outlook," *Magnetics, IEEE Transactions on,* vol. 44, no. 7, pp. 1687-1702, 2008.

[83] A. Kent, J. Yu, U. Rüdiger and S. Parkin, "Domain wall resistivity in epitaxial thin film microstructures," *Journal of Physics: Condensed Matter,* vol. 13, no. 25, p. R461, 2001.

[84] V. Martins, F. Cardoso, J. Germano, S. Cardoso, L. Sousa, M. Piedade, P. Freitas and L. Fonseca, "Femtomolar limit of detection with a magnetoresistive biochip," *Biosensors and Bioelectronics,* vol. 24, no. 8, pp. 2690-2695, 2009.

[85] C. Gooneratne, C. Liang, I. Giouroudi and J. Kosel, "An integrated micro-chip for rapid detection of magnetic particles," *Journal of Applied Physics,* vol. 111, no. 7, pp. 07B327-07B327, 2012.

[86] R. Wirix-Speetjens, G. Reekmans, R. De Palma, C. Liu, W. Laureyn and G. Borghs, "Magnetoresistive biosensors based on active guiding of magnetic particles towards the sensing zone," *Sensors and Actuators B: Chemica,* vol. 128, no. 1, pp. 1-4, 2007.

[87] Y. Nakatani, A. Thiaville and J. Miltat, "Head-to-head domain walls in soft nano-strips: a refined phase diagram," *Journal of Magnetism and Magnetic Materials,* vol. 290, pp. 750-753, 2005.

[88] "Nokia Sensing XCHALLENGE," [Online]. Available: https://sensing.xprize.org/about/overview.

[89] "Winners Announced for Competition #2," [Online]. Available: https://sensing.xprize.org/teams/competition-2-teams.

[90] "Golden Gopher Magnetic Biosensing," [Online]. Available: https://sensing.xprize.org/teams/competition-2-teams/golden-gopher-magnetic-biosensing.

[91] K. Boylan, P. Buchanan, R. Manion, D. Shukla, K. Braumberger, C. Bruggemeyer and A. Skubitz, "The expression of Nectin-4 on the surface of ovarian cancer cells alters their ability to adhere, migrate, aggregate, and proliferate," *Oncotarget,* vol. 8, no. 6, p. 9717, 2017.

[92] P. Buchanan, K. Boylan, B. Walcheck, R. Heinze, M. Geller, P. Argenta and A. Skubitz, "Ectodomain shedding of the cell adhesion molecule Nectin-4 in ovarian cancer is mediated by ADAM10 and ADAM17," *Journal of Biological Chemistry,* vol. 292, no. 15, pp. 6339-6351, 2017.

[93] Roche, "Elecsys CA-125II," [Online]. Available: http://www.cobas.com/home/product/clinical-and-immunochemistry-testing/elecsys-ca-125.html.

[94] J. Ahn, S. Choi, M. Im and others, "Charge and dielectric effects of biomolecules on electrical characteristics of nanowire FET biosensors," *Applied Physics Letters,* vol. 111, no. 11, p. 113701, 2017.

[95] J. Wang, Y. Li, C. Bian, J. Tong and others, "Ultramicroelectrode array modified with magnetically labeled Bacillus subtilis, palladium nanoparticles and reduced carboxy graphene for amperometric determination of biochemical oxygen demand," *Microchimica Acta,* vol. 184, no. 3, pp. 763-771, 2017.

[96] Center for Disease Control, "Ovarian Cancer Statistics," [Online]. Available: https://www.cdc.gov/cancer/ovarian/statistics/index.htm.

[97] Ovarian Cancer Research Fund Alliance, "Staging and Grading," 2016. [Online]. Available: https://ocrfa.org/patients/about-ovarian-cancer/treatment/staging-and-grading/. [Accessed 25 November 2017].

[98] "Magnetic Anisotropy," [Online]. Available: https://encyclopedia2.thefreedictionary.com/Magnetic+Anisotropy.

[99] "Certificate of Waiver Laboratory Project," [Online]. Available: https://www.cms.gov/Regulations-and-Guidance/Legislation/CLIA/Certificate_of_-Waiver_Laboratory_Project.html.

[100] E. Fung, "A recipe for proteomics diagnostic test development: the OVA1 test, from biomarker discovery to FDA clearance," *Clinical chemistry,* vol. 56, no. 2, pp. 327-329, 2010.

[101] T. Maggino, A. Gadducci, V. D'addario, S. Pecorelli, A. Lissoni, M. Stella, C. Romagnolo, M. Federghini, S. Zucca, D. Trio and S. Trovo, "Prospective multicenter study on CA 125 in postmenopausal pelvic masses," *Gynecologic oncology,* vol. 54, no. 2, pp. 117-123, 1994.

[102] J. Gao, L. Jeffries, K. Mach, D. Craft, N. Thomas, V. Gau, J. Liao and P. Wong, "A multiplex electrochemical biosensor for bloodstream infection diagnosis," *Journal of laboratory automation*, vol. 1, no. 2211068216651232, 2016.

[103] A. C. Society, "Survival rates for ovarian cancer, by stage," [Online]. Available: https://www.cancer.org/cancer/ovarian-cancer/detection-diagnosis-staging/survival-rates.html.

[104] W. Wang, Y. Wang, L. Tu, T. Klein, Y. Feng and J. Wang, "Surface modification for protein and DNA immobilization onto GMR biosensor," *IEEE Transactions on Magnetics*, vol. 49, no. 1, 2013.

[105] H. Zhang, T. Oellers, W. Feng and others, "High-density droplet microarray of individually addressable electrochemical cells," *Analytical Chemistry*, 2017.

Appendix A: Golden Gopher Magnetic Biosensing Team

Row 1 (Top): Jian-Ping Wang[1], Chengguo Xing[1], Amy Skubitz[1], Tian He[1], Andrew Badley[2], Michael Anderson[3], Roy Huchzermeier[4]
Row 2: Todd Klein[1], Lina Yu[1], Yi Wang[1], Yinglong Feng[1], Kai Wu[1], Jiaoming Qu[4], Lishun Wu[4]
Row 3: Michael Sandstedt[1], Jonathan Pechuman[1], Koshy Rajan[1], Amardeep Chawala[1], Liang Tu[1], Jinming Liu[1], Tim Bloomquist[5]
Row 4 (Bottom): Jim Sawyer[6], Chad Rheault[7], Bob Morke, Krishna Natarajan[1], Samiha Sultana[1], Xiqian Zheng[1], Chanqing Yin[4]

Affiliations:
1: University of Minnesota, 2: Mayo Clinic, 3: R&D Systems, 4: Zepto Life Technology, 5: Universal Magnetic Systems, 6: Jim Sawyer Professional Audio Systems, 7: Vates

Appendix B. Biochip Sensor Functionalization

The work described in this section was primarily performed by Lina Yu, a colleague in Professor Wang's research group.

Each sensor in the assay is spaced 400 μm from its neighbors. Printing different capture antibodies to individual sensors therefore required great precision and robust robotics interface. To meet the challenge, we used the Scienion S5 spotting system. The system is capable of producing droplets in the scale of hundreds of picoliters. Each individual spot illustrated in **Fig. B.1** is composed of several printed droplets and are on the scale of several nanoliters. Alignment is performed by an automatic calibration of x-y axes from images of alignment marks found on the edge of each GMR chip.

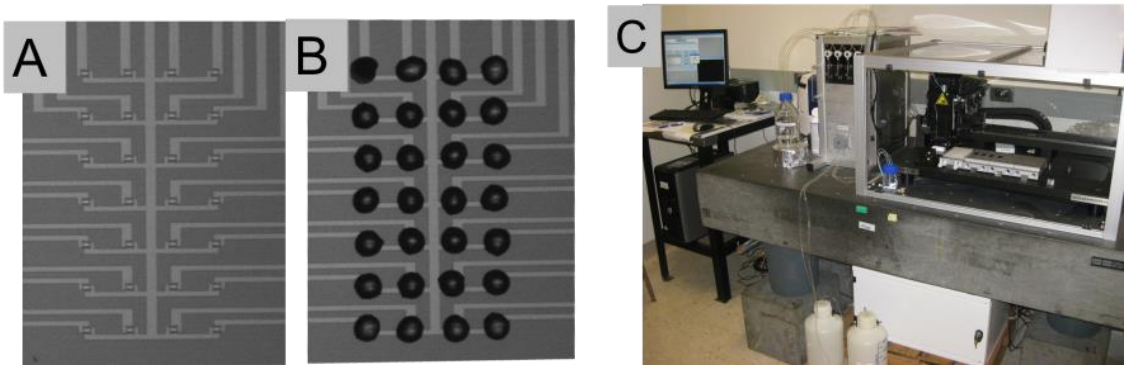

Fig. B.1. Illustrating the capability of Scienion S5 spotting system. A) GMR chip after surface modification by APTES and Glu (see below for details). B) Spots of water, each containing several printed droplets. C) Spotting system complete with robot, pumps, nozzles, cameras, and computer station.

In the development of biosensors, the immobilization of biomolecules (e. g. antibodies, antigens, nucleic acids, etc.) onto the active surface of sensors is crucial. Many functional groups, such as NHS ester and epoxy, have been deposited onto biosensor surfaces to capture proteins. However, these groups are very sensitive to water or moisture, so the storage of biosensors with functionalized surfaces for the on-site and handheld applications has been a big challenge.

We developed and demonstrated a modification method that could be applied to GMR biosensors which have a SiO_2 layer outside. In the first step, their surface was modified by 3-aminopropyltriethoxy silane (APTES), then functionalized with glutaraldehyde (Glu), resulting in the surface with aldehyde groups. IgG antibodies containing free amine groups can bind with surface aldehyde groups. The binding capabilities of the APTES-Glu modified surface and the NHS-modified surface (Surmodics) are compared in **Fig. B.2**. The results show that the modified surface using our method is almost comparable to the commercial surface.

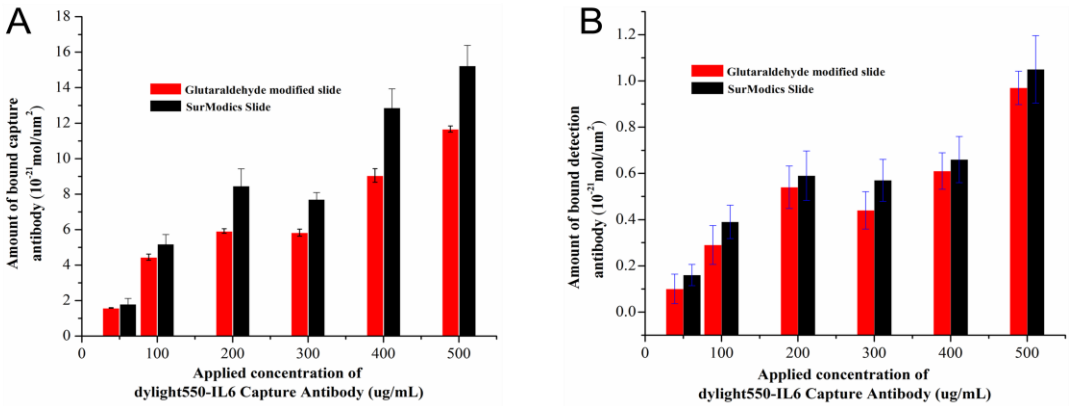

Fig. B.2. A) The efficiency of IL-6 capture antibody's immobilization on APTES-Glu modified slides and SurModics Slides were compared. **B)** The binding ability of immobilized capture antibody to IL-6 antigen and detection antibody were also Investigated. The SurModics Slides with active NHS groups on surface were provided by SurModics, Inc.

In protein binding assays, the binding of specific antibodies to specific antigens plays a significant role. Due to the complexity of protein production and their easy denaturation, the bioactivities of the proteins need to be tested each time they are purchased, as we noted in our previous publications. [19, 33] Traditionally, fluorescence methodology is employed to quantify antigen and antibody binding. An example of this is detailed in **Fig. B.3**, where IL-6 is absent (A) or present (B). The fluorescence spots were clearly detected in **Fig. B.3B**, where IL-6 is present. If this had not been the case, more detailed trouble-shooting

experiments would be required to determine the source of the problem, such protein mislabeling, denaturation of the protein, or the inappropriate selection of capture and detection antibodies.

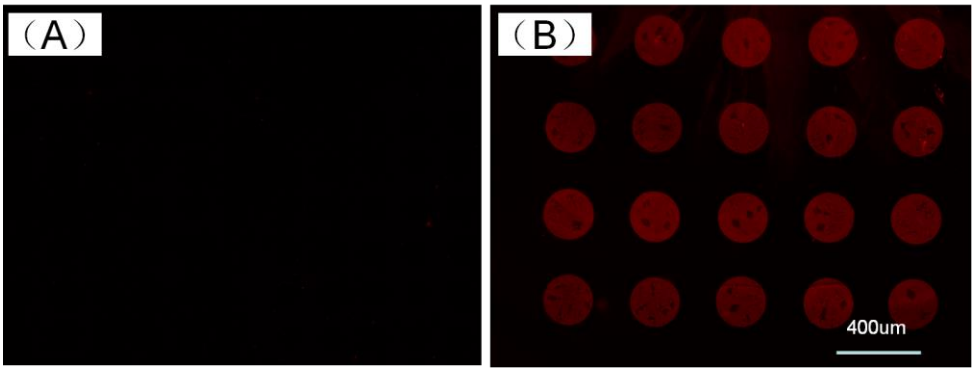

Fig. B.3. Glu-modified waters were coated with 500 ug/ml of IL-6 capture antibody, then either buffer (A) or buffer containing 3 ug/ml IL-6 (B). Biotinylated detection antibody (5 ug/mL) was added followed by AF555-Streptavidin (scale bar is 400um).

Appendix C. Copyrighted Firmware

Source Files Content

```
/*
 * asm.s
 *
 * designed and written
 * by Samiha Sultana
 *
 * First release: May 7th, 2013
 *
 */

#include "p33exxxx.h"

.global _asm16X16Mult
_asm16X16Mult:

MOV W0, W4
MOV W1, W5
MPY W4*W5, A
MOV ACCAH, W1
MOV ACCA, W0

return
.end

/*
 * balanceBridge.c
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 *
 * First release: September 12th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"
#include "spintronicsConfig.h"
#include "muxControl.h"
#include "digiPotDrv.h"
#include "uartDrv.h"
#include "fsmStates.h"
#include "commsDefines.h"
#include "utility.h"

//how close do we tune the digiPots?
#define R_AMP_MARGIN 2//out of 255
#define R_BRIDGE_MARGIN 5//out of 1535

void balanceBridgeGenerator(uint8_t runOrReset, _Q15 amplitude, _Q15 frequency, __eds__ _Q15 *cosOmegaT, __eds__ _Q15
*sinOmegaT);
void balanceBridgeMeasure(_Q15 bridgeSample, _Q15 cosOmegaT, _Q15 sinOmegaT, int64_t *cosAccumulator, int64_t
*sinAccumulator);

extern _Q15 _Q15cosPILUT(_Q15 phiOverPI);
extern _Q15 _Q15sinPILUT(_Q15 phiOverPI);

uint16_t sensorRBridgeTable[MAX_MUX_ADDRESS_TABLE_SIZE];
bool sensorRBridgeTableValid = false;

void balanceBridgeFSM(void)
{
   static uint8_t local_state;
```

```
    static _Q15 local_a1 = 0;
    static _Q15 local_f1 = 0;
    static _Q15 gain_check_measure_time;
    static _Q15 bridge_check_measure_time;
    static uint32_t timer;
    static uint8_t sensorIndex;
    _Q15 cosOmegaT;//fractions of full-scale
    _Q15 sinOmegaT;//fractions of full-scale
    static int64_t cosAccumulator;
    static int64_t sinAccumulator;

    static uint8_t r_amp;
    static uint8_t r_amp_min;
    static uint8_t r_amp_lo_mid;
    static uint8_t r_amp_hi_mid;
    static uint8_t r_amp_max;
    uint8_t r_amp_diff;
    static bool r_amp_lo_mid_clip;
    static bool r_amp_hi_mid_clip;

    static uint16_t r_bridge_min;
    static uint16_t r_bridge_lo_mid;
    static uint16_t r_bridge_hi_mid;
    static uint16_t r_bridge_max;
    uint16_t r_bridge_diff;
    static float r_bridge_lo_mid_amplitude;
    static float r_bridge_hi_mid_amplitude;

    bool advance_state = false;
    uint8_t continue_to_measurement;

    START_ATOMIC();
    continue_to_measurement = global_state & START_MEASUREMENT_AFTER_BALANCE_MASK;
    local_state = global_state & ~START_MEASUREMENT_AFTER_BALANCE_MASK;
    /*
     * clear the global state register so we can tell
     * when another request comes in via UART
     */
    global_state = 0;
    END_ATOMIC();//end critical section

    switch (local_state)
    {
       case IDLE:
          sensorIndex = 0;
          balanceBridgeGenerator(RESET_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);//use the same frequency
and amplitude to balance the bridge as will be used during measurement
          break;


       case START_BRIDGE_BALANCE_FSM:
       {

          uint32_t gain_check_measure_cycles = GAIN_CHECK_MEASURE_CYCLES;
          uint32_t gain_check_measure_time_32;
          uint32_t bridge_check_measure_time_32;

          sensorIndex = 0;

          START_ATOMIC;//begin critical section; must be atomic!
          local_a1 = bridge_balance_amplitude;
          local_f1 = bridge_balance_frequency;
          END_ATOMIC();//end critical section

          IEC3bits.DCIIE = 0;//disable the DCI interrupt in case the divide take longer than one sample period

          gain_check_measure_time_32 = gain_check_measure_cycles * 65536 / (uint16_t)local_f1;
          if (gain_check_measure_time_32 & 0xFFFF0000 != 0) {
             //gain_check_measure_time_32 will overflow the int16_t
             gain_check_measure_time = 65536 / local_f1;
          } else {
```

```
            gain_check_measure_time = gain_check_measure_time_32;
        }

        bridge_check_measure_time_32 = gain_check_measure_time_32 * BRIDGE_CHECK_MEASURE_TIME_MULTIPLIER;
        if (bridge_check_measure_time_32 & 0xFFFF0000 != 0) {
            //bridge_check_measure_time_32 will overflow the int16_t
            bridge_check_measure_time = gain_check_measure_time;
        } else {
            bridge_check_measure_time = bridge_check_measure_time_32;
        }

        IEC3bits.DCIIE = 1;//re-enable DCI interrupt

        balanceBridgeGenerator(RESET_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);//use the same frequency
and amplitude to balance the bridge as will be used during measurement
        local_state = START_GAIN_CAL;
        break;
    }


    case START_GAIN_CAL:
    {
        uint8_t sensorAddress;

        timer = 0;

        START_ATOMIC;//begin critical section; must be atomic!
        sensorAddress = sensorAddressTable[sensorIndex];
        END_ATOMIC();//end critical section

        configSensor(sensorAddress);

        //set r_bridge to initial value
        setRBridge(R_BRIDGE_MID);

        balanceBridgeGenerator(RUN_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);

        r_amp_min = R_AMP_MIN;
        r_amp_max = R_AMP_MAX;

        local_state = SET_R_AMP_TO_LO_MID;
        break;
    }

    case SET_R_AMP_TO_LO_MID:

        balanceBridgeGenerator(RUN_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);

        r_amp_lo_mid_clip = false;

        //set setRAmp to r_amp_lo_mid
        r_amp_diff = r_amp_max - r_amp_min;
        if (r_amp_diff > 7) {
            r_amp_lo_mid = r_amp_diff / 4 + r_amp_min;
        } else {
            r_amp_lo_mid = r_amp_min + 1;
        }
        setRAmp(r_amp_lo_mid);

        local_state = R_AMP_LO_MID_SIGNAL_SETTLING;
        break;


    case R_AMP_LO_MID_SIGNAL_SETTLING:

        balanceBridgeGenerator(RUN_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);

        //wait for any transients to settle
        ++timer;
        if (GAIN_CHECK_SETUP_TIME == timer) {
            timer = 0;
```

```
        local_state = R_AMP_LO_MID_CLIP_TEST;
    }
    break;


case R_AMP_LO_MID_CLIP_TEST:

    //see if the signal is clipping with this gain
    if (RXBUF0 == 0x7FFF || RXBUF0 == 0x8000) {
        r_amp_lo_mid_clip = true;
        advance_state = true;
    }

    balanceBridgeGenerator(RUN_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);

    ++timer;
    if (advance_state || gain_check_measure_time == timer) {
        timer = 0;
        local_state = SET_R_AMP_TO_HI_MID;
    }
    break;

case SET_R_AMP_TO_HI_MID:

    balanceBridgeGenerator(RUN_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);
    r_amp_hi_mid_clip = false;

    //set setRAmp to r_amp_hi_mid
    r_amp_diff = r_amp_max - r_amp_min;
    if (r_amp_diff > 2) {
        r_amp_hi_mid = (uint16_t)r_amp_diff * 3 / 4 + r_amp_min;
    } else {
        r_amp_hi_mid = r_amp_max - 1;
    }
    setRAmp(r_amp_hi_mid);

    local_state = R_AMP_HI_MID_SIGNAL_SETTLING;
    break;

case R_AMP_HI_MID_SIGNAL_SETTLING:

    balanceBridgeGenerator(RUN_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);

    //wait for any transients to settle
    ++timer;
    if (GAIN_CHECK_SETUP_TIME == timer) {
        timer = 0;
        local_state = R_AMP_HI_MID_CLIP_TEST;
    }
    break;

case R_AMP_HI_MID_CLIP_TEST:

    //see if the signal is clipping with this gain
    if (RXBUF0 == 0x7FFF || RXBUF0 == 0x8000) {
        r_amp_hi_mid_clip = true;
        advance_state = true;
    }

    balanceBridgeGenerator(RUN_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);

    ++timer;
    if (advance_state || gain_check_measure_time == timer) {
        timer = 0;

        if (!r_amp_lo_mid_clip && !r_amp_hi_mid_clip) {
            r_amp_min = r_amp_lo_mid;
        } else if (r_amp_lo_mid_clip) {
            r_amp_max = r_amp_lo_mid;
        } else {
            r_amp_min = r_amp_lo_mid;
```

```
            r_amp_max = r_amp_hi_mid;
        }

        if (   (r_amp_min > r_amp_max)
            || (r_amp_max - r_amp_min < R_AMP_MARGIN)) {

            if (r_amp_min > R_AMP_MARGIN) {
                r_amp = r_amp_min - R_AMP_MARGIN;
                setRAmp(r_amp);
            } else {
                r_amp = 0;
                setRAmp(r_amp);
            }

            local_state = START_BRIDGE_CAL;

        } else {

            local_state = SET_R_AMP_TO_LO_MID;

        }
    }
    break;


case START_BRIDGE_CAL:

    balanceBridgeGenerator(RESET_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);

    r_bridge_min = R_BRIDGE_MIN;
    r_bridge_max = R_BRIDGE_MAX;

    local_state = SET_R_BRIDGE_TO_LO_MID;

    break;


case SET_R_BRIDGE_TO_LO_MID:

    balanceBridgeGenerator(RESET_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);

    cosAccumulator = 0;
    sinAccumulator = 0;

    //set set RBridge to r_bridge_lo_mid
    r_bridge_diff = r_bridge_max - r_bridge_min;
    if (r_bridge_diff > 7) {
        r_bridge_lo_mid = r_bridge_diff / 4 + r_bridge_min;
    } else {
        r_bridge_lo_mid = r_bridge_min + 1;
    }
    setRBridge(r_bridge_lo_mid);

    local_state = R_BRIDGE_LO_MID_SIGNAL_SETTLING;
    break;


case R_BRIDGE_LO_MID_SIGNAL_SETTLING:

    balanceBridgeGenerator(RUN_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);

    //wait for any transients to settle
    ++timer;
    if (BRIDGE_CHECK_SETUP_TIME == timer) {
        timer = 0;
        local_state = R_BRIDGE_LO_MID_AMP_MEASURE;
    }
    break;

case R_BRIDGE_LO_MID_AMP_MEASURE:
{
```

```
        volatile _Q15 bridgeSample;

        bridgeSample = RXBUF0;

        //see if the signal is clipping with this gain
        if (bridgeSample == 0x7FFF || bridgeSample == 0x8000) {

            //we clipped! reduce gain and abort this measurement
            timer = 0;
            if (0 == r_amp) {
                //bridge amp is at minimum gain; calibration has failed
                local_state = IDLE;
                transmitError(BRIDGE_BALANCE_FAILURE);
            } else if (r_amp > R_AMP_MARGIN) {
                r_amp -= R_AMP_MARGIN;
                setRAmp(r_amp);
                local_state = START_BRIDGE_CAL;
            } else {
                r_amp = 0;
                setRAmp(r_amp);
                local_state = START_BRIDGE_CAL;
            }

        } else {

            balanceBridgeMeasure(bridgeSample, cosOmegaT, sinOmegaT, &cosAccumulator, &sinAccumulator);
            ++timer;
            if (bridge_check_measure_time == timer) {
                timer = 0;
                local_state = R_BRIDGE_LO_MID_AMP_CALC;
            } else {
                balanceBridgeGenerator(RUN_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);
            }

        }
        break;
    }


    case R_BRIDGE_LO_MID_AMP_CALC:
    {
        double cosAccumulatorFloat;
        double sinAccumulatorFloat;
        double phaseAngle;

        IEC3bits.DCIIE = 0;//disable the interrupt while this calculation is taking place because the floating point calculations take
much longer than one sample period

        cosAccumulatorFloat = (double)cosAccumulator;
        sinAccumulatorFloat = (double)sinAccumulator;

        phaseAngle = atan2(sinAccumulatorFloat, cosAccumulatorFloat);

        if (fabs(cosAccumulatorFloat) > fabs(sinAccumulatorFloat))
        {
            //NOTE: units are arbitrary!
            r_bridge_lo_mid_amplitude = cosAccumulatorFloat / cosf(phaseAngle);
        }
        else
        {
            //NOTE: units are arbitrary!
            r_bridge_lo_mid_amplitude = sinAccumulatorFloat / sinf(phaseAngle);
        }

        if(r_bridge_lo_mid_amplitude < 0)//make sure amplitude is positive
        {
            r_bridge_lo_mid_amplitude = fabsf(r_bridge_lo_mid_amplitude);
        }

        local_state = SET_R_BRIDGE_TO_HI_MID;
        IEC3bits.DCIIE = 1;//re-enable the DCI interrupt
```

```
    break;
}


case SET_R_BRIDGE_TO_HI_MID:

    balanceBridgeGenerator(RESET_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);

    cosAccumulator = 0;
    sinAccumulator = 0;

    //set set RBridge to r_bridge_hi_mid
    r_bridge_diff = r_bridge_max - r_bridge_min;
    if (r_bridge_diff > 2) {
        r_bridge_hi_mid = r_bridge_diff * 3 / 4 + r_bridge_min;
    } else {
        r_bridge_hi_mid = r_bridge_max - 1;
    }
    setRBridge(r_bridge_hi_mid);

    local_state = R_BRIDGE_HI_MID_SIGNAL_SETTLING;
    break;


case R_BRIDGE_HI_MID_SIGNAL_SETTLING:

    balanceBridgeGenerator(RUN_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);

    //wait for any transients to settle
    ++timer;
    if (BRIDGE_CHECK_SETUP_TIME == timer) {
        timer = 0;
        local_state = R_BRIDGE_HI_MID_AMP_MEASURE;
    }
    break;

case R_BRIDGE_HI_MID_AMP_MEASURE:
{
    volatile _Q15 bridgeSample;

    bridgeSample = RXBUF0;

    //see if the signal is clipping with this gain
    if (bridgeSample == 0x7FFF || bridgeSample == 0x8000) {

        //we clipped! reduce gain and abort this measurement
        timer = 0;
        if (0 == r_amp) {
            //bridge amp is at minimum gain; calibration has failed
            local_state = IDLE;
            transmitError(BRIDGE_BALANCE_FAILURE);
        } else if (r_amp > R_AMP_MARGIN) {
            r_amp -= R_AMP_MARGIN;
            setRAmp(r_amp);
            local_state = START_BRIDGE_CAL;
        } else {
            r_amp = 0;
            setRAmp(r_amp);
            local_state = START_BRIDGE_CAL;
        }

    } else {

        balanceBridgeMeasure(bridgeSample, cosOmegaT, sinOmegaT, &cosAccumulator, &sinAccumulator);
        ++timer;
        if (bridge_check_measure_time == timer) {
            timer = 0;
            local_state = R_BRIDGE_HI_MID_AMP_CALC;
        } else {
            balanceBridgeGenerator(RUN_SIGNAL_GEN, local_a1, local_f1, &cosOmegaT, &sinOmegaT);
        }
```

```
        }
        break;
    }


    case R_BRIDGE_HI_MID_AMP_CALC:
    {
        double cosAccumulatorFloat;
        double sinAccumulatorFloat;
        double phaseAngle;
        uint16_t cal_window;

        IEC3bits.DCIIE = 0;//disable the interrupt while this calculation is taking place because the floating point calculations take much longer than one sample period

        cosAccumulatorFloat = (double)cosAccumulator;
        sinAccumulatorFloat = (double)sinAccumulator;

        phaseAngle = atan2(sinAccumulatorFloat, cosAccumulatorFloat);

        if (fabs(cosAccumulatorFloat) > fabs(sinAccumulatorFloat))
        {
            //NOTE: units are arbitrary!
            r_bridge_hi_mid_amplitude = cosAccumulatorFloat / cosf(phaseAngle);
        }
        else
        {
            //NOTE: units are arbitrary!
            r_bridge_hi_mid_amplitude = sinAccumulatorFloat / sinf(phaseAngle);
        }

        if(r_bridge_hi_mid_amplitude < 0)//make sure amplitude is positive
        {
            r_bridge_hi_mid_amplitude = fabsf(r_bridge_lo_mid_amplitude);
        }

        if (r_bridge_lo_mid_amplitude > r_bridge_hi_mid_amplitude) {
            r_bridge_min = r_bridge_lo_mid;
        } else {
            r_bridge_max = r_bridge_hi_mid;
        }

        if (r_bridge_min < r_bridge_max) {
            cal_window = r_bridge_max - r_bridge_min;
        } else {
            cal_window = r_bridge_min - r_bridge_max;
        }

        if (cal_window <= R_BRIDGE_MARGIN) {

            sensorRBridgeTable[sensorIndex] = r_bridge_min;
            ++sensorIndex;
            START_ATOMIC();
            if (sensorIndex >= numberOfSensors) {

                /*
                 * must test >= in case numberOfSensors was updated since
                 * the last iteration of this FSM
                 */
                sensorRBridgeTableValid = true;
                if (continue_to_measurement) {
                    local_state = START_MEASUREMENT_FSM;
                    continue_to_measurement = 0;
                } else {
                    local_state = IDLE;
                }

            } else {
                local_state = START_GAIN_CAL;
            }
```

```
            } else {

                local_state = SET_R_BRIDGE_TO_LO_MID;

            }

            IEC3bits.DCIIE = 1;//re-enable the DCI interrupt
            break;
        }

        default:
            local_state = IDLE;
            break;

    }

    START_ATOMIC();//begin critical section; must be atomic!
    if (!global_state) {
        /*
         * global_state register is clear; no pending requests have arrived over
         * UART so it's safe to write in our new state
         */
        global_state = local_state | continue_to_measurement;

    }
    END_ATOMIC();//end critical section

}

void balanceBridgeGenerator(uint8_t runOrReset, _Q15 amplitude, _Q15 frequency, __eds__ _Q15 *cosOmegaT, __eds__ _Q15
*sinOmegaT)
{
    static _Q15 freqT;
    if (runOrReset == RESET_SIGNAL_GEN)
    {
        TXBUF0 = 0x0000;
        TXBUF1 = 0x0000;
        TXBUF2 = 0x0000;
        TXBUF3 = 0x0000;

        freqT = 0;
    }
    else
    {
        *cosOmegaT = _Q15cosPILUT(freqT);//generating cos(omega * t)
        *sinOmegaT = _Q15sinPILUT(freqT);//generating sin(omega * t)

        _Q31_t tempSample;
        if (amplitude == 0x7FFF)
        {
            TXBUF0 = *cosOmegaT;
            TXBUF1 = 0x0000;
        }
        else
        {
            tempSample.Q31 = asm16X16Mult(*cosOmegaT, a1);
            TXBUF0 = tempSample.dword.highWord;
            TXBUF1 = tempSample.dword.lowWord;
        }
#ifndef PROBE
        TXBUF2 = 0x0000;
        TXBUF3 = 0x0000;
#elif defined(PROBE_BRIDGE_ADC)
        TXBUF2 = RXBUF0;
        TXBUF3 = RXBUF1;
#elif defined(PROBE_COIL_ADC)
        TXBUF2 = RXBUF2;
        TXBUF3 = RXBUF3;
#endif
        freqT += frequency;
```

```
    }
}
void balanceBridgeMeasure(_Q15 bridgeSample, _Q15 cosOmegaT, _Q15 sinOmegaT, int64_t *cosAccumulator, int64_t
*sinAccumulator)
{
    _Q31_t bridgeByCosFT, bridgeBySinFT;

    bridgeByCosFT.Q31 = asm16X16Mult(bridgeSample, cosOmegaT);
    *cosAccumulator += bridgeByCosFT.Q31;

    bridgeBySinFT.Q31 = asm16X16Mult(bridgeSample, sinOmegaT);
    *sinAccumulator += bridgeBySinFT.Q31;
}

/*
 * calculateVectors.c
 *
 * Author: Michael Reinhart Sandstedt
 *
 * First release: September 14th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"
#include "spintronicsConfig.h"
#include "constants.h"
#include "uartDrv.h"
#include "spintronicsStructs.h"
#include "calculateVectors.h"

#define M_PI 3.14159265358979323846

//global variables
uint8_t sensorAddressCapture;
Q31_IQ_array_t IQ_capture;
bool bridgeADCClipCapture;
bool coilADCClipCapture;
bool bridgeDigitalClipCapture;
bool f1PlusF2OutOfRangeCapture;
float inverseBridgeAnalogGainCapture;

static float
calculatePhase(_Q31_t I, _Q31_t Q)
{
    return atan2((double)(Q.Q31), (double)(I.Q31));
}

static float
calculateNormalizedAmplitude(_Q31_t I, _Q31_t Q)
{
    return sqrt((double)(I.Q31) * (double)(I.Q31) + (double)(Q.Q31) * (double)(Q.Q31));
}

void calculateFinalVectors(void)
{
    float_array_t amplitude;
    float_array_t phase;

    phase.bridge_f1 = calculatePhase(IQ_capture.f1_I, IQ_capture.f1_Q);
    amplitude.bridge_f1 = calculateNormalizedAmplitude(IQ_capture.f1_I, IQ_capture.f1_Q);
#ifdef MEASURE_F2_AT_BRIDGE
    phase.bridge_f2 = calculatePhase(IQ_capture.bridge_f2_I, IQ_capture.bridge_f2_Q);
    amplitude.bridge_f2 = calculateNormalizedAmplitude(IQ_capture.bridge_f2_I, IQ_capture.bridge_f2_Q);
#endif
    phase.bridge_fdiff = calculatePhase(IQ_capture.fdiff_I, IQ_capture.fdiff_Q);
    amplitude.bridge_fdiff = calculateNormalizedAmplitude(IQ_capture.fdiff_I, IQ_capture.fdiff_Q);
    phase.bridge_fsum = calculatePhase(IQ_capture.fsum_I, IQ_capture.fsum_Q);
    amplitude.bridge_fsum = calculateNormalizedAmplitude(IQ_capture.fsum_I, IQ_capture.fsum_Q);
#ifdef MEASURE_F2_AT_COIL
```

```
   phase.coil_f2 = calculatePhase(IQ_capture.coil_f2_I, IQ_capture.coil_f2_Q);
   amplitude.coil_f2 = calculateNormalizedAmplitude(IQ_capture.coil_f2_I, IQ_capture.coil_f2_Q);
#endif

   switch(bridgeADCGainFactor)//if digital gain was added, need to scale the reported voltage appropriately
   {
      case 0:

         amplitude.bridge_f1 = amplitude.bridge_f1 * BRIDGE_ADC_SCALE_FACTOR * inverseBridgeAnalogGainCapture;

         #ifdef MEASURE_F2_AT_BRIDGE
         amplitude.bridge_f2 = amplitude.bridge_f2 * BRIDGE_ADC_SCALE_FACTOR * inverseBridgeAnalogGainCapture;
         #endif

         amplitude.bridge_fdiff = amplitude.bridge_fdiff * BRIDGE_ADC_SCALE_FACTOR * inverseBridgeAnalogGainCapture;
         if (f1PlusF2OutOfRangeCapture)
         {
            amplitude.bridge_fsum = 0;
         }
         else
         {
            amplitude.bridge_fsum = amplitude.bridge_fsum * BRIDGE_ADC_SCALE_FACTOR *
inverseBridgeAnalogGainCapture;
         }
         break;


      case 1:

         amplitude.bridge_f1 = amplitude.bridge_f1 * BRIDGE_ADC_SCALE_FACTOR_BY_2 * inverseBridgeAnalogGainCapture;

         #ifdef MEASURE_F2_AT_BRIDGE
         amplitude.bridge_f2 = amplitude.bridge_f2 * BRIDGE_ADC_SCALE_FACTOR_BY_2 * inverseBridgeAnalogGainCapture;
         #endif

         amplitude.bridge_fdiff = amplitude.bridge_fdiff * BRIDGE_ADC_SCALE_FACTOR_BY_2 *
inverseBridgeAnalogGainCapture;
         if (f1PlusF2OutOfRangeCapture)
         {
            amplitude.bridge_fsum = 0;
         }
         else
         {
            amplitude.bridge_fsum = amplitude.bridge_fsum * BRIDGE_ADC_SCALE_FACTOR_BY_2 *
inverseBridgeAnalogGainCapture;
         }
         break;


      case 2:

         amplitude.bridge_f1 = amplitude.bridge_f1 * BRIDGE_ADC_SCALE_FACTOR_BY_4 * inverseBridgeAnalogGainCapture;

         #ifdef MEASURE_F2_AT_BRIDGE
         amplitude.bridge_f2 = amplitude.bridge_f2 * BRIDGE_ADC_SCALE_FACTOR_BY_4 * inverseBridgeAnalogGainCapture;
         #endif

         amplitude.bridge_fdiff = amplitude.bridge_fdiff * BRIDGE_ADC_SCALE_FACTOR_BY_4 *
inverseBridgeAnalogGainCapture;
         if (f1PlusF2OutOfRangeCapture)
         {
            amplitude.bridge_fsum = 0;
         }
         else
         {
            amplitude.bridge_fsum = amplitude.bridge_fsum * BRIDGE_ADC_SCALE_FACTOR_BY_4 *
inverseBridgeAnalogGainCapture;
         }
         break;
```

```
      case 3:

         amplitude.bridge_f1 = amplitude.bridge_f1 * BRIDGE_ADC_SCALE_FACTOR_BY_8 * inverseBridgeAnalogGainCapture;

         #ifdef MEASURE_F2_AT_BRIDGE
         amplitude.bridge_f2 = amplitude.bridge_f2 * BRIDGE_ADC_SCALE_FACTOR_BY_8 * inverseBridgeAnalogGainCapture;
         #endif

         amplitude.bridge_fdiff = amplitude.bridge_fdiff * BRIDGE_ADC_SCALE_FACTOR_BY_8 *
inverseBridgeAnalogGainCapture;
         if (f1PlusF2OutOfRangeCapture)
         {
            amplitude.bridge_fsum = 0;
         }
         else
         {
            amplitude.bridge_fsum = amplitude.bridge_fsum * BRIDGE_ADC_SCALE_FACTOR_BY_8 *
inverseBridgeAnalogGainCapture;
         }
         break;


      case 4:

         amplitude.bridge_f1 = amplitude.bridge_f1 * BRIDGE_ADC_SCALE_FACTOR_BY_16 *
inverseBridgeAnalogGainCapture;

         #ifdef MEASURE_F2_AT_BRIDGE
         amplitude.bridge_f2 = amplitude.bridge_f2 * BRIDGE_ADC_SCALE_FACTOR_BY_16 *
inverseBridgeAnalogGainCapture;
         #endif

         amplitude.bridge_fdiff = amplitude.bridge_fdiff * BRIDGE_ADC_SCALE_FACTOR_BY_16 *
inverseBridgeAnalogGainCapture;
         if (f1PlusF2OutOfRangeCapture)
         {
            amplitude.bridge_fsum = 0;
         }
         else
         {
            amplitude.bridge_fsum = amplitude.bridge_fsum * BRIDGE_ADC_SCALE_FACTOR_BY_16 *
inverseBridgeAnalogGainCapture;
         }
         break;


      default:

         amplitude.bridge_f1 = amplitude.bridge_f1 * BRIDGE_ADC_SCALE_FACTOR * inverseBridgeAnalogGainCapture;

         #ifdef MEASURE_F2_AT_BRIDGE
         amplitude.bridge_f2 = amplitude.bridge_f2 * BRIDGE_ADC_SCALE_FACTOR * inverseBridgeAnalogGainCapture;
         #endif

         amplitude.bridge_fdiff = amplitude.bridge_fdiff * BRIDGE_ADC_SCALE_FACTOR * inverseBridgeAnalogGainCapture;
         if (f1PlusF2OutOfRangeCapture)
         {
            amplitude.bridge_fsum = 0;
         }
         else
         {
            amplitude.bridge_fsum = amplitude.bridge_fsum * BRIDGE_ADC_SCALE_FACTOR *
inverseBridgeAnalogGainCapture;
         }
         break;

   }

#ifdef MESSURE_F2_AT_COIL
   amplitude.coil_f2 = amplitude.coil_f2 * COIL_ADC_SCALE_FACTOR;
#endif
```

```
    transmitResults(sensorAddressCapture, &phase, &amplitude,
                bridgeADCClipCapture, coilADCClipCapture,
                bridgeDigitalClipCapture);

}


/*
 * cosLUT.c
 *
 * this code and all concepts contained
 * within designed and written
 * by Michael Reinhart Sandstedt
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"
#include "spintronicsConfig.h"
#include "spintronicsStructs.h"

/*
 * contents of the array defined as follows:
 * cosLUT[i] =  (int16_t)(cos(pi * (_Q15)(i + 1) * 32768)
 *  */

__prog__ _Q15 __attribute__((space(prog)))
cosLUT_ZeroToPiByTwo[0x3FFFu] = {32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767,
32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767,
32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767,
32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767,
32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32767, 32766, 32766,
32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766,
32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32766, 32765, 32765, 32765, 32765, 32765,
32765, 32765, 32765, 32765, 32765, 32765, 32765, 32765, 32765, 32765, 32765, 32765, 32765, 32765, 32765, 32765, 32765,
32765, 32765, 32765, 32764, 32764, 32764, 32764, 32764, 32764, 32764, 32764, 32764, 32764, 32764, 32764, 32764, 32764,
32764, 32764, 32764, 32764, 32764, 32764, 32763, 32763, 32763, 32763, 32763, 32763, 32763, 32763, 32763, 32763, 32763,
32763, 32763, 32763, 32763, 32763, 32763, 32763, 32763, 32762, 32762, 32762, 32762, 32762, 32762, 32762, 32762, 32762,
32762, 32762, 32762, 32762, 32762, 32762, 32762, 32761, 32761, 32761, 32761, 32761, 32761, 32761, 32761, 32761, 32761,
32761, 32761, 32761, 32761, 32761, 32760, 32760, 32760, 32760, 32760, 32760, 32760, 32760, 32760, 32760, 32760, 32760,
32760, 32760, 32759, 32759, 32759, 32759, 32759, 32759, 32759, 32759, 32759, 32759, 32759, 32759, 32759, 32759, 32758,
32758, 32758, 32758, 32758, 32758, 32758, 32758, 32758, 32758, 32758, 32757, 32757, 32757, 32757, 32757, 32757, 32757,
32757, 32757, 32757, 32757, 32757, 32757, 32756, 32756, 32756, 32756, 32756, 32756, 32756, 32756, 32756, 32756, 32756,
32755, 32755, 32755, 32755, 32755, 32755, 32755, 32755, 32755, 32755, 32755, 32754, 32754, 32754, 32754, 32754, 32754,
32754, 32754, 32754, 32754, 32754, 32753, 32753, 32753, 32753, 32753, 32753, 32753, 32753, 32753, 32753, 32752, 32752,
32752, 32752, 32752, 32752, 32752, 32752, 32752, 32751, 32751, 32751, 32751, 32751, 32751, 32751, 32751, 32751, 32751,
32750, 32750, 32750, 32750, 32750, 32750, 32750, 32750, 32750, 32749, 32749, 32749, 32749, 32749, 32749, 32749, 32749,
32749, 32748, 32748, 32748, 32748, 32748, 32748, 32748, 32748, 32748, 32747, 32747, 32747, 32747, 32747, 32747, 32747,
32747, 32746, 32746, 32746, 32746, 32746, 32746, 32746, 32746, 32746, 32745, 32745, 32745, 32745, 32745, 32745, 32745,
32744, 32744, 32744, 32744, 32744, 32744, 32744, 32744, 32744, 32743, 32743, 32743, 32743, 32743, 32743, 32743, 32742,
32742, 32742, 32742, 32742, 32742, 32742, 32742, 32741, 32741, 32741, 32741, 32741, 32741, 32741, 32741, 32740, 32740,
32740, 32740, 32740, 32740, 32740, 32739, 32739, 32739, 32739, 32739, 32739, 32739, 32738, 32738, 32738, 32738, 32738,
32738, 32738, 32738, 32737, 32737, 32737, 32737, 32737, 32737, 32737, 32736, 32736, 32736, 32736, 32736, 32736, 32736,
32735, 32735, 32735, 32735, 32735, 32735, 32735, 32734, 32734, 32734, 32734, 32734, 32734, 32734, 32733, 32733, 32733,
32733, 32733, 32733, 32732, 32732, 32732, 32732, 32732, 32732, 32731, 32731, 32731, 32731, 32731, 32731, 32731, 32730,
32730, 32730, 32730, 32730, 32730, 32730, 32729, 32729, 32729, 32729, 32729, 32729, 32729, 32728, 32728, 32728, 32728,
32728, 32728, 32728, 32727, 32727, 32727, 32727, 32727, 32727, 32726, 32726, 32726, 32726, 32726, 32726, 32726, 32725,
32725, 32725, 32725, 32725, 32725, 32725, 32724, 32724, 32724, 32724, 32724, 32724, 32724, 32723, 32723, 32723, 32723,
32723, 32723, 32723, 32722, 32722, 32722, 32722, 32722, 32722, 32722, 32721, 32721, 32721, 32721, 32721, 32721, 32720,
32720, 32720, 32720, 32720, 32720, 32720, 32719, 32719, 32719, 32719, 32719, 32719, 32719, 32718, 32718, 32718, 32718,
32718, 32718, 32718, 32717, 32717, 32717, 32717, 32717, 32717, 32717, 32716, 32716, 32716, 32716, 32716, 32716, 32715,
32715, 32715, 32715, 32715, 32715, 32715, 32714, 32714, 32714, 32714, 32714, 32714, 32713, 32713, 32713, 32713, 32713,
32713, 32713, 32712, 32712, 32712, 32712, 32712, 32712, 32712, 32711, 32711, 32711, 32711, 32711, 32711, 32711, 32710,
32710, 32710, 32710, 32710, 32710, 32710, 32709, 32709, 32709, 32709, 32709, 32709, 32709, 32708, 32708, 32708, 32708,
32708, 32708, 32708, 32707, 32707, 32707, 32707, 32707, 32707, 32707, 32706, 32706, 32706, 32706, 32706, 32706, 32706,
32705, 32705, 32705, 32705, 32705, 32705, 32705, 32704, 32704, 32704, 32704, 32704, 32704, 32704, 32703, 32703, 32703,
32703, 32703, 32703, 32703, 32702, 32702, 32702, 32702, 32702, 32702, 32702, 32701, 32701, 32701, 32701, 32701, 32701,
32701, 32700, 32700, 32700, 32700, 32700, 32700, 32700, 32699, 32699, 32699, 32699, 32699, 32699, 32699, 32698, 32698,
32698, 32698, 32698, 32698, 32698, 32697, 32697, 32697, 32697, 32697, 32697, 32697, 32696, 32696, 32696, 32696, 32696,
32696, 32696, 32695, 32695, 32695, 32695, 32695, 32695, 32695, 32694, 32694, 32694, 32694, 32694, 32694, 32694, 32693,
32693, 32693, 32693, 32693, 32693, 32693, 32692, 32692, 32692, 32692, 32692, 32692, 32692, 32691, 32691, 32691, 32691,
```

32691, 32690, 32690, 32690, 32690, 32689, 32689, 32689, 32689, 32689, 32688, 32688, 32688, 32688, 32687, 32687, 32687, 32687,
32687, 32686, 32686, 32686, 32686, 32686, 32685, 32685, 32685, 32685, 32684, 32684, 32684, 32684, 32684, 32683, 32683, 32683,
32683, 32682, 32682, 32682, 32682, 32681, 32681, 32681, 32681, 32681, 32680, 32680, 32680, 32680, 32679, 32679, 32679, 32679,
32678, 32678, 32678, 32678, 32678, 32677, 32677, 32677, 32677, 32676, 32676, 32676, 32676, 32675, 32675, 32675, 32675, 32675,
32674, 32674, 32674, 32674, 32673, 32673, 32673, 32673, 32672, 32672, 32672, 32672, 32671, 32671, 32671, 32671, 32670, 32670,
32670, 32670, 32669, 32669, 32669, 32669, 32668, 32668, 32668, 32668, 32668, 32667, 32667, 32667, 32667, 32666, 32666, 32666,
32666, 32665, 32665, 32665, 32665, 32664, 32664, 32664, 32664, 32663, 32663, 32663, 32663, 32662, 32662, 32662, 32662, 32661,
32661, 32661, 32661, 32660, 32660, 32660, 32660, 32659, 32659, 32659, 32658, 32658, 32658, 32658, 32657, 32657, 32657, 32657,
32656, 32656, 32656, 32656, 32655, 32655, 32655, 32655, 32654, 32654, 32654, 32654, 32653, 32653, 32653, 32653, 32652, 32652,
32652, 32651, 32651, 32651, 32651, 32650, 32650, 32650, 32650, 32649, 32649, 32649, 32649, 32648, 32648, 32648, 32647, 32647,
32647, 32647, 32646, 32646, 32646, 32646, 32645, 32645, 32645, 32645, 32644, 32644, 32644, 32643, 32643, 32643, 32643, 32642,
32642, 32642, 32641, 32641, 32641, 32641, 32640, 32640, 32640, 32640, 32639, 32639, 32639, 32638, 32638, 32638, 32638, 32637,
32637, 32637, 32636, 32636, 32636, 32636, 32635, 32635, 32635, 32635, 32634, 32634, 32634, 32633, 32633, 32633, 32633, 32632,
32632, 32632, 32631, 32631, 32631, 32631, 32630, 32630, 32630, 32629, 32629, 32629, 32629, 32628, 32628, 32628, 32627, 32627,
32627, 32626, 32626, 32626, 32626, 32625, 32625, 32625, 32624, 32624, 32624, 32624, 32623, 32623, 32623, 32622, 32622, 32622,
32621, 32621, 32621, 32621, 32620, 32620, 32620, 32619, 32619, 32619, 32619, 32618, 32618, 32618, 32617, 32617, 32617, 32616,
32616, 32616, 32616, 32615, 32615, 32615, 32614, 32614, 32614, 32613, 32613, 32613, 32612, 32612, 32612, 32612, 32611, 32611,
32611, 32610, 32610, 32610, 32609, 32609, 32609, 32608, 32608, 32608, 32608, 32607, 32607, 32607, 32606, 32606, 32606, 32605,
32605, 32605, 32604, 32604, 32604, 32603, 32603, 32603, 32603, 32602, 32602, 32602, 32601, 32601, 32601, 32600, 32600, 32600,
32599, 32599, 32599, 32598, 32598, 32598, 32597, 32597, 32597, 32597, 32596, 32596, 32596, 32595, 32595, 32595, 32594, 32594,
32594, 32593, 32593, 32593, 32592, 32592, 32592, 32591, 32591, 32591, 32590, 32590, 32590, 32589, 32589, 32589, 32588, 32588,
32588, 32587, 32587, 32587, 32586, 32586, 32586, 32585, 32585, 32585, 32584, 32584, 32584, 32583, 32583, 32583, 32582, 32582,
32582, 32581, 32581, 32581, 32580, 32580, 32580, 32579, 32579, 32579, 32578, 32578, 32578, 32577, 32577, 32577, 32576, 32576,
32576, 32575, 32575, 32575, 32574, 32574, 32574, 32573, 32573, 32573, 32572, 32572, 32572, 32571, 32571, 32571, 32570, 32570,
32570, 32569, 32569, 32569, 32568, 32568, 32567, 32567, 32567, 32566, 32566, 32566, 32565, 32565, 32565, 32564, 32564, 32564,
32563, 32563, 32563, 32562, 32562, 32562, 32561, 32561, 32561, 32560, 32560, 32559, 32559, 32559, 32558, 32558, 32558, 32557,
32557, 32557, 32556, 32556, 32556, 32555, 32555, 32554, 32554, 32554, 32553, 32553, 32553, 32552, 32552, 32552, 32551, 32551,
32551, 32550, 32550, 32549, 32549, 32548, 32548, 32548, 32547, 32547, 32547, 32546, 32546, 32545, 32545, 32545, 32544, 32544,
32544, 32544, 32543, 32543, 32543, 32542, 32542, 32541, 32541, 32541, 32540, 32540, 32540, 32539, 32539, 32538, 32538, 32538,
32537, 32537, 32537, 32536, 32536, 32535, 32535, 32535, 32534, 32534, 32534, 32533, 32533, 32532, 32532, 32532, 32531, 32531,
32531, 32530, 32530, 32529, 32529, 32529, 32528, 32528, 32528, 32527, 32527, 32526, 32526, 32526, 32525, 32525, 32525, 32524,
32524, 32523, 32523, 32523, 32522, 32522, 32521, 32521, 32521, 32520, 32520, 32520, 32519, 32519, 32518, 32518, 32518, 32517,
32517, 32516, 32516, 32516, 32515, 32515, 32514, 32514, 32514, 32513, 32513, 32513, 32512, 32512, 32511, 32511, 32511, 32510,
32510, 32509, 32509, 32509, 32508, 32508, 32507, 32507, 32507, 32506, 32506, 32505, 32505, 32505, 32504, 32504, 32503, 32503,
32503, 32502, 32502, 32501, 32501, 32501, 32500, 32500, 32499, 32499, 32499, 32498, 32498, 32497, 32497, 32497, 32496, 32496,
32495, 32495, 32495, 32494, 32494, 32493, 32493, 32493, 32492, 32492, 32491, 32491, 32491, 32490, 32490, 32489, 32489, 32489,
32488, 32488, 32487, 32487, 32486, 32486, 32486, 32485, 32485, 32484, 32484, 32484, 32483, 32483, 32482, 32482, 32482, 32481,
32481, 32480, 32480, 32479, 32479, 32479, 32478, 32478, 32477, 32477, 32477, 32476, 32476, 32475, 32475, 32474, 32474, 32474,
32473, 32473, 32472, 32472, 32472, 32471, 32471, 32470, 32470, 32469, 32469, 32469, 32468, 32468, 32467, 32467, 32466, 32466,
32466, 32465, 32465, 32464, 32464, 32463, 32463, 32463, 32462, 32462, 32461, 32461, 32460, 32460, 32460, 32459, 32459, 32458,
32458, 32457, 32457, 32457, 32456, 32456, 32455, 32455, 32454, 32454, 32454, 32453, 32453, 32452, 32452, 32451, 32451, 32451,
32450, 32450, 32449, 32449, 32448, 32448, 32447, 32447, 32447, 32446, 32446, 32445, 32445, 32444, 32444, 32443, 32443, 32443,
32442, 32442, 32441, 32441, 32440, 32440, 32440, 32439, 32439, 32438, 32438, 32437, 32437, 32436, 32436, 32436, 32435, 32435,
32434, 32434, 32433, 32433, 32432, 32432, 32431, 32431, 32431, 32430, 32430, 32429, 32429, 32428, 32428, 32427, 32427, 32427,
32426, 32426, 32425, 32425, 32424, 32424, 32423, 32423, 32422, 32422, 32422, 32421, 32421, 32420, 32420, 32419, 32419, 32418,
32418, 32417, 32417, 32417, 32416, 32416, 32415, 32415, 32414, 32414, 32413, 32413, 32412, 32412, 32411, 32411, 32411, 32410,
32410, 32409, 32409, 32408, 32408, 32407, 32407, 32406, 32406, 32405, 32405, 32404, 32404, 32404, 32403, 32403, 32402, 32402,
32401, 32401, 32400, 32400, 32399, 32399, 32398, 32398, 32397, 32397, 32397, 32396, 32396, 32395, 32395, 32394, 32394, 32393,
32393, 32392, 32392, 32391, 32391, 32390, 32390, 32389, 32389, 32388, 32388, 32388, 32387, 32387, 32386, 32386, 32385, 32385,
32384, 32384, 32383, 32383, 32382, 32382, 32381, 32381, 32380, 32380, 32379, 32379, 32378, 32378, 32377, 32377, 32376, 32376,
32375, 32375, 32375, 32374, 32374, 32373, 32373, 32372, 32372, 32371, 32371, 32370, 32370, 32369, 32369, 32368, 32368, 32367,
32367, 32366, 32366, 32365, 32365, 32364, 32364, 32363, 32363, 32362, 32362, 32361, 32361, 32360, 32360, 32359, 32359, 32358,
32358, 32357, 32357, 32356, 32356, 32355, 32355, 32354, 32354, 32353, 32353, 32352, 32352, 32351, 32351, 32350, 32350, 32349,
32349, 32348, 32348, 32347, 32347, 32346, 32346, 32345, 32345, 32344, 32344, 32343, 32343, 32342, 32342, 32341, 32341, 32340,
32340, 32339, 32339, 32338, 32338, 32337, 32337, 32336, 32336, 32335, 32335, 32334, 32334, 32333, 32333, 32332, 32332, 32331,
32331, 32330, 32330, 32329, 32329, 32328, 32328, 32327, 32327, 32326, 32326, 32325, 32325, 32324, 32323, 32323, 32322, 32322,
32321, 32321, 32320, 32320, 32319, 32319, 32318, 32318, 32317, 32317, 32316, 32316, 32315, 32315, 32314, 32314, 32313, 32313,
32312, 32312, 32311, 32311, 32310, 32309, 32309, 32308, 32308, 32307, 32307, 32306, 32306, 32305, 32305, 32304, 32304, 32303,
32303, 32302, 32302, 32301, 32301, 32300, 32299, 32299, 32298, 32298, 32297, 32297, 32296, 32296, 32295, 32295, 32294, 32294,
32293, 32293, 32292, 32292, 32291, 32290, 32290, 32289, 32289, 32288, 32288, 32287, 32287, 32286, 32286, 32285, 32285, 32284,
32283, 32283, 32282, 32282, 32281, 32281, 32280, 32280, 32279, 32279, 32278, 32278, 32277, 32276, 32276, 32275, 32275, 32274,
32274, 32273, 32273, 32272, 32272, 32271, 32270, 32270, 32269, 32269, 32268, 32268, 32267, 32267, 32266, 32266, 32265, 32264,
32264, 32263, 32263, 32262, 32262, 32261, 32261, 32260, 32260, 32259, 32258, 32258, 32257, 32257, 32256, 32256, 32255, 32255,
32254, 32253, 32253, 32252, 32252, 32251, 32251, 32250, 32250, 32249, 32248, 32248, 32247, 32247, 32246, 32246, 32245, 32245,
32244, 32243, 32243, 32242, 32242, 32241, 32241, 32240, 32239, 32239, 32238, 32238, 32237, 32237, 32236, 32236, 32235, 32234,
32234, 32233, 32233, 32232, 32232, 32231, 32230, 32230, 32229, 32229, 32228, 32228, 32227, 32226, 32226, 32225, 32225, 32224,
32224, 32223, 32222, 32222, 32221, 32221, 32220, 32220, 32219, 32218, 32218, 32217, 32217, 32216, 32216, 32215, 32214, 32214,
32213, 32213, 32212, 32212, 32211, 32210, 32210, 32209, 32209, 32208, 32208, 32207, 32206, 32206, 32205, 32205, 32204, 32204,
32203, 32202, 32202, 32201, 32201, 32200, 32199, 32199, 32198, 32198, 32197, 32197, 32196, 32195, 32195, 32194, 32194, 32193,
32192, 32192, 32191, 32191, 32190, 32189, 32189, 32188, 32188, 32187, 32187, 32186, 32185, 32185, 32184, 32184, 32183, 32182,
32182, 32181, 32181, 32180, 32179, 32179, 32178, 32178, 32177, 32176, 32176, 32175, 32175, 32174, 32174, 32173, 32172, 32172,

32171, 32171, 32170, 32169, 32169, 32168, 32168, 32167, 32166, 32166, 32165, 32165, 32164, 32163, 32163, 32162, 32162, 32161,
32160, 32160, 32159, 32159, 32158, 32157, 32157, 32156, 32156, 32155, 32154, 32154, 32153, 32152, 32152, 32151, 32151, 32150,
32149, 32149, 32148, 32148, 32147, 32146, 32146, 32145, 32145, 32144, 32143, 32143, 32142, 32142, 32141, 32140, 32140, 32139,
32138, 32138, 32137, 32137, 32136, 32135, 32135, 32134, 32134, 32133, 32132, 32132, 32131, 32130, 32130, 32129, 32129, 32128,
32127, 32127, 32126, 32126, 32125, 32124, 32124, 32123, 32122, 32122, 32121, 32121, 32120, 32119, 32119, 32118, 32117, 32117,
32116, 32116, 32115, 32114, 32114, 32113, 32112, 32112, 32111, 32111, 32110, 32109, 32109, 32108, 32107, 32107, 32106, 32106,
32105, 32104, 32104, 32103, 32102, 32102, 32101, 32101, 32100, 32099, 32099, 32098, 32097, 32097, 32096, 32096, 32095, 32094,
32094, 32093, 32092, 32092, 32091, 32090, 32090, 32089, 32089, 32088, 32087, 32087, 32086, 32085, 32085, 32084, 32083, 32083,
32082, 32082, 32081, 32080, 32080, 32079, 32078, 32078, 32077, 32076, 32076, 32075, 32074, 32074, 32073, 32073, 32072, 32071,
32071, 32070, 32069, 32069, 32068, 32067, 32067, 32066, 32065, 32065, 32064, 32063, 32063, 32062, 32062, 32061, 32060, 32060,
32059, 32058, 32058, 32057, 32056, 32056, 32055, 32054, 32054, 32053, 32052, 32052, 32051, 32050, 32050, 32049, 32049, 32048,
32047, 32047, 32046, 32045, 32045, 32044, 32043, 32043, 32042, 32041, 32041, 32040, 32039, 32039, 32038, 32037, 32037, 32036,
32035, 32035, 32034, 32033, 32033, 32032, 32031, 32031, 32030, 32029, 32029, 32028, 32027, 32027, 32026, 32025, 32025, 32024,
32023, 32023, 32022, 32021, 32021, 32020, 32019, 32019, 32018, 32017, 32017, 32016, 32015, 32015, 32014, 32013, 32013, 32012,
32011, 32011, 32010, 32009, 32009, 32008, 32007, 32007, 32006, 32005, 32005, 32004, 32003, 32003, 32002, 32001, 32001, 32000,
31999, 31999, 31998, 31997, 31997, 31996, 31995, 31995, 31994, 31993, 31993, 31992, 31991, 31990, 31990, 31989, 31988, 31988,
31987, 31986, 31986, 31985, 31984, 31984, 31983, 31982, 31982, 31981, 31980, 31980, 31979, 31978, 31977, 31977, 31976, 31975,
31975, 31974, 31973, 31973, 31972, 31971, 31971, 31970, 31969, 31969, 31968, 31967, 31966, 31966, 31965, 31964, 31964, 31963,
31962, 31962, 31961, 31960, 31960, 31959, 31958, 31957, 31957, 31956, 31955, 31955, 31954, 31953, 31953, 31952, 31951, 31951,
31950, 31949, 31948, 31948, 31947, 31946, 31946, 31945, 31944, 31944, 31943, 31942, 31941, 31941, 31940, 31939, 31939, 31938,
31937, 31937, 31936, 31935, 31934, 31934, 31933, 31932, 31932, 31931, 31930, 31929, 31929, 31928, 31927, 31927, 31926, 31925,
31925, 31924, 31923, 31922, 31922, 31921, 31920, 31920, 31919, 31918, 31917, 31917, 31916, 31915, 31915, 31914, 31913, 31912,
31912, 31911, 31910, 31910, 31909, 31908, 31907, 31907, 31906, 31905, 31905, 31904, 31903, 31902, 31902, 31901, 31900, 31900,
31899, 31898, 31897, 31897, 31896, 31895, 31895, 31894, 31893, 31892, 31892, 31891, 31890, 31889, 31889, 31888, 31887, 31887,
31886, 31885, 31884, 31884, 31883, 31882, 31882, 31881, 31880, 31879, 31879, 31878, 31877, 31876, 31876, 31875, 31874, 31874,
31873, 31872, 31871, 31871, 31870, 31869, 31868, 31868, 31867, 31866, 31865, 31865, 31864, 31863, 31863, 31862, 31861, 31860,
31860, 31859, 31858, 31857, 31857, 31856, 31855, 31854, 31854, 31853, 31852, 31852, 31851, 31850, 31849, 31849, 31848, 31847,
31846, 31846, 31845, 31844, 31843, 31843, 31842, 31841, 31840, 31840, 31839, 31838, 31837, 31837, 31836, 31835, 31834, 31834,
31833, 31832, 31832, 31831, 31830, 31829, 31829, 31828, 31827, 31826, 31826, 31825, 31824, 31823, 31823, 31822, 31821, 31820,
31820, 31819, 31818, 31817, 31817, 31816, 31815, 31814, 31814, 31813, 31812, 31811, 31811, 31810, 31809, 31808, 31807, 31807,
31806, 31805, 31804, 31804, 31803, 31802, 31801, 31801, 31800, 31799, 31798, 31798, 31797, 31796, 31795, 31795, 31794, 31793,
31792, 31792, 31791, 31790, 31789, 31789, 31788, 31787, 31786, 31785, 31785, 31784, 31783, 31782, 31782, 31781, 31780, 31779,
31779, 31778, 31777, 31776, 31776, 31775, 31774, 31773, 31772, 31772, 31771, 31770, 31769, 31769, 31768, 31767, 31766, 31766,
31765, 31764, 31763, 31762, 31762, 31761, 31760, 31759, 31759, 31758, 31757, 31756, 31755, 31755, 31754, 31753, 31752, 31752,
31751, 31750, 31749, 31749, 31748, 31747, 31746, 31745, 31745, 31744, 31743, 31742, 31741, 31741, 31740, 31739, 31738, 31738,
31737, 31736, 31735, 31734, 31734, 31733, 31732, 31731, 31731, 31730, 31729, 31728, 31727, 31727, 31726, 31725, 31724, 31723,
31723, 31722, 31721, 31720, 31720, 31719, 31718, 31717, 31716, 31716, 31715, 31714, 31713, 31712, 31712, 31711, 31710, 31709,
31708, 31708, 31707, 31706, 31705, 31705, 31704, 31703, 31702, 31701, 31701, 31700, 31699, 31698, 31697, 31697, 31696, 31695,
31694, 31693, 31693, 31692, 31691, 31690, 31689, 31689, 31688, 31687, 31686, 31685, 31685, 31684, 31683, 31682, 31681, 31681,
31680, 31679, 31678, 31677, 31677, 31676, 31675, 31674, 31673, 31673, 31672, 31671, 31670, 31669, 31669, 31668, 31667, 31666,
31665, 31664, 31664, 31663, 31662, 31661, 31660, 31660, 31659, 31658, 31657, 31656, 31656, 31655, 31654, 31653, 31652, 31651,
31651, 31650, 31649, 31648, 31647, 31647, 31646, 31645, 31644, 31643, 31643, 31642, 31641, 31640, 31639, 31638, 31638, 31637,
31636, 31635, 31634, 31634, 31633, 31632, 31631, 31630, 31629, 31629, 31628, 31627, 31626, 31625, 31625, 31624, 31623, 31622,
31621, 31620, 31620, 31619, 31618, 31617, 31616, 31615, 31615, 31614, 31613, 31612, 31611, 31610, 31610, 31609, 31608, 31607,
31606, 31606, 31605, 31604, 31603, 31602, 31601, 31601, 31600, 31599, 31598, 31597, 31596, 31596, 31595, 31594, 31593, 31592,
31591, 31591, 31590, 31589, 31588, 31587, 31586, 31586, 31585, 31584, 31583, 31582, 31581, 31581, 31580, 31579, 31578, 31577,
31576, 31575, 31575, 31574, 31573, 31572, 31571, 31570, 31570, 31569, 31568, 31567, 31566, 31565, 31565, 31564, 31563, 31562,
31561, 31560, 31559, 31559, 31558, 31557, 31556, 31555, 31554, 31554, 31553, 31552, 31551, 31550, 31549, 31548, 31548, 31547,
31546, 31545, 31544, 31543, 31543, 31542, 31541, 31540, 31539, 31538, 31537, 31537, 31536, 31535, 31534, 31533, 31532, 31531,
31531, 31530, 31529, 31528, 31527, 31526, 31525, 31525, 31524, 31523, 31522, 31521, 31520, 31519, 31519, 31518, 31517, 31516,
31515, 31514, 31513, 31513, 31512, 31511, 31510, 31509, 31508, 31507, 31507, 31506, 31505, 31504, 31503, 31502, 31501, 31500,
31500, 31499, 31498, 31497, 31496, 31495, 31494, 31494, 31493, 31492, 31491, 31490, 31489, 31488, 31487, 31487, 31486, 31485,
31484, 31483, 31482, 31481, 31480, 31480, 31479, 31478, 31477, 31476, 31475, 31474, 31474, 31473, 31472, 31471, 31470, 31469,
31468, 31467, 31467, 31466, 31465, 31464, 31463, 31462, 31461, 31460, 31459, 31459, 31458, 31457, 31456, 31455, 31454, 31453,
31452, 31452, 31451, 31450, 31449, 31448, 31447, 31446, 31445, 31445, 31444, 31443, 31442, 31441, 31440, 31439, 31438, 31437,
31437, 31436, 31435, 31434, 31433, 31432, 31431, 31430, 31429, 31429, 31428, 31427, 31426, 31425, 31424, 31423, 31422, 31421,
31421, 31420, 31419, 31418, 31417, 31416, 31415, 31414, 31413, 31413, 31412, 31411, 31410, 31409, 31408, 31407, 31406, 31405,
31404, 31404, 31403, 31402, 31401, 31400, 31399, 31398, 31397, 31396, 31395, 31395, 31394, 31393, 31392, 31391, 31390, 31389,
31388, 31387, 31386, 31386, 31385, 31384, 31383, 31382, 31381, 31380, 31379, 31378, 31377, 31377, 31376, 31375, 31374, 31373,
31372, 31371, 31370, 31369, 31368, 31367, 31367, 31366, 31365, 31364, 31363, 31362, 31361, 31360, 31359, 31358, 31357, 31357,
31356, 31355, 31354, 31353, 31352, 31351, 31350, 31349, 31348, 31347, 31346, 31346, 31345, 31344, 31343, 31342, 31341, 31340,
31339, 31338, 31337, 31336, 31335, 31335, 31334, 31333, 31332, 31331, 31330, 31329, 31328, 31327, 31326, 31325, 31324, 31324,
31323, 31322, 31321, 31320, 31319, 31318, 31317, 31316, 31315, 31314, 31313, 31312, 31311, 31311, 31310, 31309, 31308, 31307,
31306, 31305, 31304, 31303, 31302, 31301, 31300, 31299, 31298, 31298, 31297, 31296, 31295, 31294, 31293, 31292, 31291, 31290,
31289, 31288, 31287, 31286, 31285, 31285, 31284, 31283, 31282, 31281, 31280, 31279, 31278, 31277, 31276, 31275, 31274, 31273,
31272, 31271, 31270, 31270, 31269, 31268, 31267, 31266, 31265, 31264, 31263, 31262, 31261, 31260, 31259, 31258, 31257, 31256,
31255, 31254, 31254, 31253, 31252, 31251, 31250, 31249, 31248, 31247, 31246, 31245, 31244, 31243, 31242, 31241, 31240, 31239,
31238, 31237, 31236, 31236, 31235, 31234, 31233, 31232, 31231, 31230, 31229, 31228, 31227, 31226, 31225, 31224, 31223, 31222,
31221, 31220, 31219, 31218, 31217, 31216, 31216, 31215, 31214, 31213, 31212, 31211, 31210, 31209, 31208, 31207, 31206, 31205,
31204, 31203, 31202, 31201, 31200, 31199, 31198, 31197, 31196, 31195, 31194, 31193, 31193, 31192, 31191, 31190, 31189, 31188,
31187, 31186, 31185, 31184, 31183, 31182, 31181, 31180, 31179, 31178, 31177, 31176, 31175, 31174, 31173, 31172, 31171, 31170,

31169, 31168, 31167, 31166, 31165, 31164, 31164, 31163, 31162, 31161, 31160, 31159, 31158, 31157, 31156, 31155, 31154, 31153,
31152, 31151, 31150, 31149, 31148, 31147, 31146, 31145, 31144, 31143, 31142, 31141, 31140, 31139, 31138, 31137, 31136, 31135,
31134, 31133, 31132, 31131, 31130, 31129, 31128, 31127, 31126, 31125, 31124, 31123, 31122, 31122, 31121, 31120, 31119, 31118,
31117, 31116, 31115, 31114, 31113, 31112, 31111, 31110, 31109, 31108, 31107, 31106, 31105, 31104, 31103, 31102, 31101, 31100,
31099, 31098, 31097, 31096, 31095, 31094, 31093, 31092, 31091, 31090, 31089, 31088, 31087, 31086, 31085, 31084, 31083, 31082,
31081, 31080, 31079, 31078, 31077, 31076, 31075, 31074, 31073, 31072, 31071, 31070, 31069, 31068, 31067, 31066, 31065, 31064,
31063, 31062, 31061, 31060, 31059, 31058, 31057, 31056, 31055, 31054, 31053, 31052, 31051, 31050, 31049, 31048, 31047, 31046,
31045, 31044, 31043, 31042, 31041, 31040, 31039, 31038, 31037, 31036, 31035, 31034, 31033, 31032, 31031, 31030, 31029, 31028,
31027, 31026, 31025, 31024, 31023, 31022, 31021, 31020, 31019, 31018, 31017, 31016, 31015, 31014, 31013, 31012, 31011, 31010,
31009, 31008, 31007, 31006, 31005, 31004, 31002, 31001, 31000, 30999, 30998, 30997, 30996, 30995, 30994, 30993, 30992, 30991,
30990, 30989, 30988, 30987, 30986, 30985, 30984, 30983, 30982, 30981, 30980, 30979, 30978, 30977, 30976, 30975, 30974, 30973,
30972, 30971, 30970, 30969, 30968, 30967, 30966, 30965, 30964, 30963, 30962, 30961, 30960, 30958, 30957, 30956, 30955, 30954,
30953, 30952, 30951, 30950, 30949, 30948, 30947, 30946, 30945, 30944, 30943, 30942, 30941, 30940, 30939, 30938, 30937, 30936,
30935, 30934, 30933, 30932, 30931, 30930, 30929, 30927, 30926, 30925, 30924, 30923, 30922, 30921, 30920, 30919, 30918, 30917,
30916, 30915, 30914, 30913, 30912, 30911, 30910, 30909, 30908, 30907, 30906, 30905, 30904, 30902, 30901, 30900, 30899, 30898,
30897, 30896, 30895, 30894, 30893, 30892, 30891, 30890, 30889, 30888, 30887, 30886, 30885, 30884, 30883, 30882, 30880, 30879,
30878, 30877, 30876, 30875, 30874, 30873, 30872, 30871, 30870, 30869, 30868, 30867, 30866, 30865, 30864, 30863, 30862, 30860,
30859, 30858, 30857, 30856, 30855, 30854, 30853, 30852, 30851, 30850, 30849, 30848, 30847, 30846, 30845, 30844, 30842, 30841,
30840, 30839, 30838, 30837, 30836, 30835, 30834, 30833, 30832, 30831, 30830, 30829, 30828, 30827, 30825, 30824, 30823, 30822,
30821, 30820, 30819, 30818, 30817, 30816, 30815, 30814, 30813, 30812, 30811, 30809, 30808, 30807, 30806, 30805, 30804, 30803,
30802, 30801, 30800, 30799, 30798, 30797, 30796, 30794, 30793, 30792, 30791, 30790, 30789, 30788, 30787, 30786, 30785, 30784,
30783, 30782, 30780, 30779, 30778, 30777, 30776, 30775, 30774, 30773, 30772, 30771, 30770, 30769, 30768, 30766, 30765, 30764,
30763, 30762, 30761, 30760, 30759, 30758, 30757, 30756, 30755, 30753, 30752, 30751, 30750, 30749, 30748, 30747, 30746, 30745,
30744, 30743, 30742, 30740, 30739, 30738, 30737, 30736, 30735, 30734, 30733, 30732, 30731, 30730, 30728, 30727, 30726, 30725,
30724, 30723, 30722, 30721, 30720, 30719, 30718, 30716, 30715, 30714, 30713, 30712, 30711, 30710, 30709, 30708, 30707, 30705,
30704, 30703, 30702, 30701, 30700, 30699, 30698, 30697, 30696, 30694, 30693, 30692, 30691, 30690, 30689, 30688, 30687, 30686,
30685, 30683, 30682, 30681, 30680, 30679, 30678, 30677, 30676, 30675, 30674, 30672, 30671, 30670, 30669, 30668, 30667, 30666,
30665, 30664, 30662, 30661, 30660, 30659, 30658, 30657, 30656, 30655, 30654, 30652, 30651, 30650, 30649, 30648, 30647, 30646,
30645, 30644, 30642, 30641, 30640, 30639, 30638, 30637, 30636, 30635, 30634, 30632, 30631, 30630, 30629, 30628, 30627, 30626,
30625, 30624, 30622, 30621, 30620, 30619, 30618, 30617, 30616, 30615, 30613, 30612, 30611, 30610, 30609, 30608, 30607, 30606,
30604, 30603, 30602, 30601, 30600, 30599, 30598, 30597, 30595, 30594, 30593, 30592, 30591, 30590, 30589, 30588, 30586, 30585,
30584, 30583, 30582, 30581, 30580, 30579, 30577, 30576, 30575, 30574, 30573, 30572, 30571, 30570, 30568, 30567, 30566, 30565,
30564, 30563, 30562, 30560, 30559, 30558, 30557, 30556, 30555, 30554, 30553, 30551, 30550, 30549, 30548, 30547, 30546, 30545,
30543, 30542, 30541, 30540, 30539, 30538, 30537, 30535, 30534, 30533, 30532, 30531, 30530, 30529, 30527, 30526, 30525, 30524,
30523, 30522, 30521, 30520, 30518, 30517, 30516, 30515, 30514, 30513, 30511, 30510, 30509, 30508, 30507, 30506, 30505, 30503,
30502, 30501, 30500, 30499, 30498, 30497, 30495, 30494, 30493, 30492, 30491, 30490, 30489, 30487, 30486, 30485, 30484, 30483,
30482, 30480, 30479, 30478, 30477, 30476, 30475, 30474, 30472, 30471, 30470, 30469, 30468, 30467, 30465, 30464, 30463, 30462,
30461, 30460, 30458, 30457, 30456, 30455, 30454, 30453, 30452, 30450, 30449, 30448, 30447, 30446, 30445, 30443, 30442, 30441,
30440, 30439, 30438, 30436, 30435, 30434, 30433, 30432, 30431, 30429, 30428, 30427, 30426, 30425, 30424, 30422, 30421, 30420,
30419, 30418, 30417, 30415, 30414, 30413, 30412, 30411, 30410, 30408, 30407, 30406, 30405, 30404, 30403, 30401, 30400, 30399,
30398, 30397, 30396, 30394, 30393, 30392, 30391, 30390, 30388, 30387, 30386, 30385, 30384, 30383, 30381, 30380, 30379, 30378,
30377, 30376, 30374, 30373, 30372, 30371, 30370, 30368, 30367, 30366, 30365, 30364, 30363, 30361, 30360, 30359, 30358, 30357,
30355, 30354, 30353, 30352, 30351, 30350, 30348, 30347, 30346, 30345, 30344, 30342, 30341, 30340, 30339, 30338, 30337, 30335,
30334, 30333, 30332, 30331, 30329, 30328, 30327, 30326, 30325, 30323, 30322, 30321, 30320, 30319, 30317, 30316, 30315, 30314,
30313, 30312, 30310, 30309, 30308, 30307, 30306, 30304, 30303, 30302, 30301, 30300, 30298, 30297, 30296, 30295, 30294, 30292,
30291, 30290, 30289, 30288, 30286, 30285, 30284, 30283, 30282, 30280, 30279, 30278, 30277, 30276, 30274, 30273, 30272, 30271,
30270, 30268, 30267, 30266, 30265, 30264, 30262, 30261, 30260, 30259, 30258, 30256, 30255, 30254, 30253, 30251, 30250, 30249,
30248, 30247, 30245, 30244, 30243, 30242, 30241, 30239, 30238, 30237, 30236, 30235, 30233, 30232, 30231, 30230, 30229, 30227,
30226, 30225, 30224, 30222, 30221, 30220, 30219, 30218, 30216, 30215, 30214, 30213, 30212, 30210, 30209, 30208, 30207, 30205,
30204, 30203, 30202, 30201, 30199, 30198, 30197, 30196, 30194, 30193, 30192, 30191, 30190, 30188, 30187, 30186, 30185, 30183,
30182, 30181, 30180, 30179, 30177, 30176, 30175, 30174, 30172, 30171, 30170, 30169, 30168, 30166, 30165, 30164, 30163, 30161,
30160, 30159, 30158, 30156, 30155, 30154, 30153, 30152, 30150, 30149, 30148, 30147, 30145, 30144, 30143, 30142, 30140, 30139,
30138, 30137, 30136, 30134, 30133, 30132, 30131, 30129, 30128, 30127, 30126, 30124, 30123, 30122, 30121, 30119, 30118, 30117,
30116, 30115, 30113, 30112, 30111, 30110, 30108, 30107, 30106, 30105, 30103, 30102, 30101, 30100, 30098, 30097, 30096, 30095,
30093, 30092, 30091, 30090, 30088, 30087, 30086, 30085, 30084, 30082, 30081, 30080, 30079, 30077, 30076, 30075, 30074, 30072,
30071, 30070, 30069, 30067, 30066, 30065, 30064, 30062, 30061, 30060, 30059, 30057, 30056, 30055, 30054, 30052, 30051, 30050,
30049, 30047, 30046, 30045, 30044, 30042, 30041, 30040, 30038, 30037, 30036, 30035, 30033, 30032, 30031, 30030, 30028, 30027,
30026, 30025, 30023, 30022, 30021, 30020, 30018, 30017, 30016, 30015, 30013, 30012, 30011, 30010, 30008, 30007, 30006, 30005,
30003, 30002, 30001, 29999, 29998, 29997, 29996, 29994, 29993, 29992, 29991, 29989, 29988, 29987, 29986, 29984, 29983, 29982,
29980, 29979, 29978, 29977, 29975, 29974, 29973, 29972, 29970, 29969, 29968, 29967, 29965, 29964, 29963, 29961, 29960, 29959,
29958, 29956, 29955, 29954, 29953, 29951, 29950, 29949, 29947, 29946, 29945, 29944, 29942, 29941, 29940, 29938, 29937, 29936,
29935, 29933, 29932, 29931, 29930, 29928, 29927, 29926, 29924, 29923, 29922, 29921, 29919, 29918, 29917, 29915, 29914, 29913,
29912, 29910, 29909, 29908, 29906, 29905, 29904, 29903, 29901, 29900, 29899, 29897, 29896, 29895, 29894, 29892, 29891, 29890,
29888, 29887, 29886, 29885, 29883, 29882, 29881, 29879, 29878, 29877, 29876, 29874, 29873, 29872, 29870, 29869, 29868, 29867,
29865, 29864, 29863, 29861, 29860, 29859, 29857, 29856, 29855, 29854, 29852, 29851, 29850, 29848, 29847, 29846, 29845, 29843,
29842, 29841, 29839, 29838, 29837, 29835, 29834, 29833, 29832, 29830, 29829, 29828, 29826, 29825, 29824, 29822, 29821, 29820,
29819, 29817, 29816, 29815, 29813, 29812, 29811, 29809, 29808, 29807, 29805, 29804, 29803, 29802, 29800, 29799, 29798, 29796,
29795, 29794, 29792, 29791, 29790, 29788, 29787, 29786, 29785, 29783, 29782, 29781, 29779, 29778, 29777, 29775, 29774, 29773,
29771, 29770, 29769, 29768, 29766, 29765, 29764, 29762, 29761, 29760, 29758, 29757, 29756, 29754, 29753, 29752, 29750, 29749,
29748, 29746, 29745, 29744, 29743, 29741, 29740, 29739, 29737, 29736, 29735, 29733, 29732, 29731, 29729, 29728, 29727, 29725,
29724, 29723, 29721, 29720, 29719, 29717, 29716, 29715, 29713, 29712, 29711, 29709, 29708, 29707, 29706, 29704, 29703, 29702,

29700, 29699, 29698, 29696, 29695, 29694, 29692, 29691, 29690, 29688, 29687, 29686, 29684, 29683, 29682, 29680, 29679, 29678,
29676, 29675, 29674, 29672, 29671, 29670, 29668, 29667, 29666, 29664, 29663, 29662, 29660, 29659, 29658, 29656, 29655, 29654,
29652, 29651, 29650, 29648, 29647, 29646, 29644, 29643, 29642, 29640, 29639, 29638, 29636, 29635, 29633, 29632, 29631, 29629,
29628, 29627, 29625, 29624, 29623, 29621, 29620, 29619, 29617, 29616, 29615, 29613, 29612, 29611, 29609, 29608, 29607, 29605,
29604, 29603, 29601, 29600, 29599, 29597, 29596, 29595, 29593, 29592, 29590, 29589, 29588, 29586, 29585, 29584, 29582, 29581,
29580, 29578, 29577, 29576, 29574, 29573, 29572, 29570, 29569, 29567, 29566, 29565, 29563, 29562, 29561, 29559, 29558, 29557,
29555, 29554, 29553, 29551, 29550, 29548, 29547, 29546, 29544, 29543, 29542, 29540, 29539, 29538, 29536, 29535, 29534, 29532,
29531, 29529, 29528, 29527, 29525, 29524, 29523, 29521, 29520, 29519, 29517, 29516, 29514, 29513, 29512, 29510, 29509, 29508,
29506, 29505, 29504, 29502, 29501, 29499, 29498, 29497, 29495, 29494, 29493, 29491, 29490, 29488, 29487, 29486, 29484, 29483,
29482, 29480, 29479, 29478, 29476, 29475, 29473, 29472, 29471, 29469, 29468, 29467, 29465, 29464, 29462, 29461, 29460, 29458,
29457, 29456, 29454, 29453, 29451, 29450, 29449, 29447, 29446, 29445, 29443, 29442, 29440, 29439, 29438, 29436, 29435, 29433,
29432, 29431, 29429, 29428, 29427, 29425, 29424, 29422, 29421, 29420, 29418, 29417, 29416, 29414, 29413, 29411, 29410, 29409,
29407, 29406, 29404, 29403, 29402, 29400, 29399, 29397, 29396, 29395, 29393, 29392, 29391, 29389, 29388, 29386, 29385, 29384,
29382, 29381, 29379, 29378, 29377, 29375, 29374, 29372, 29371, 29370, 29368, 29367, 29365, 29364, 29363, 29361, 29360, 29359,
29357, 29356, 29354, 29353, 29352, 29350, 29349, 29347, 29346, 29345, 29343, 29342, 29340, 29339, 29338, 29336, 29335, 29333,
29332, 29331, 29329, 29328, 29326, 29325, 29324, 29322, 29321, 29319, 29318, 29317, 29315, 29314, 29312, 29311, 29310, 29308,
29307, 29305, 29304, 29302, 29301, 29300, 29298, 29297, 29295, 29294, 29293, 29291, 29290, 29288, 29287, 29286, 29284, 29283,
29281, 29280, 29279, 29277, 29276, 29274, 29273, 29271, 29270, 29269, 29267, 29266, 29264, 29263, 29262, 29260, 29259, 29257,
29256, 29255, 29253, 29252, 29250, 29249, 29247, 29246, 29245, 29243, 29242, 29240, 29239, 29238, 29236, 29235, 29233, 29232,
29230, 29229, 29228, 29226, 29225, 29223, 29222, 29220, 29219, 29218, 29216, 29215, 29213, 29212, 29211, 29209, 29208, 29206,
29205, 29203, 29202, 29201, 29199, 29198, 29196, 29195, 29193, 29192, 29191, 29189, 29188, 29186, 29185, 29183, 29182, 29181,
29179, 29178, 29176, 29175, 29173, 29172, 29171, 29169, 29168, 29166, 29165, 29163, 29162, 29161, 29159, 29158, 29156, 29155,
29153, 29152, 29151, 29149, 29148, 29146, 29145, 29143, 29142, 29140, 29139, 29138, 29136, 29135, 29133, 29132, 29130, 29129,
29128, 29126, 29125, 29123, 29122, 29120, 29119, 29117, 29116, 29115, 29113, 29112, 29110, 29109, 29107, 29106, 29104, 29103,
29102, 29100, 29099, 29097, 29096, 29094, 29093, 29091, 29090, 29089, 29087, 29086, 29084, 29083, 29081, 29080, 29078, 29077,
29076, 29074, 29073, 29071, 29070, 29068, 29067, 29065, 29064, 29062, 29061, 29060, 29058, 29057, 29055, 29054, 29052, 29051,
29049, 29048, 29047, 29045, 29044, 29042, 29041, 29039, 29038, 29036, 29035, 29033, 29032, 29031, 29029, 29028, 29026, 29025,
29023, 29022, 29020, 29019, 29017, 29016, 29014, 29013, 29012, 29010, 29009, 29007, 29006, 29004, 29003, 29001, 29000, 28998,
28997, 28995, 28994, 28993, 28991, 28990, 28988, 28987, 28985, 28984, 28982, 28981, 28979, 28978, 28976, 28975, 28973, 28972,
28971, 28969, 28968, 28966, 28965, 28963, 28962, 28960, 28959, 28957, 28956, 28954, 28953, 28951, 28950, 28948, 28947, 28946,
28944, 28943, 28941, 28940, 28938, 28937, 28935, 28934, 28932, 28931, 28929, 28928, 28926, 28925, 28923, 28922, 28920, 28919,
28918, 28916, 28915, 28913, 28912, 28910, 28909, 28907, 28906, 28904, 28903, 28901, 28900, 28898, 28897, 28895, 28894, 28892,
28891, 28889, 28888, 28886, 28885, 28883, 28882, 28881, 28879, 28878, 28876, 28875, 28873, 28872, 28870, 28869, 28867, 28866,
28864, 28863, 28861, 28860, 28858, 28857, 28855, 28854, 28852, 28851, 28849, 28848, 28846, 28845, 28843, 28842, 28840, 28839,
28837, 28836, 28834, 28833, 28831, 28830, 28828, 28827, 28825, 28824, 28822, 28821, 28819, 28818, 28816, 28815, 28813, 28812,
28810, 28809, 28807, 28806, 28804, 28803, 28801, 28800, 28798, 28797, 28795, 28794, 28792, 28791, 28789, 28788, 28786, 28785,
28783, 28782, 28780, 28779, 28777, 28776, 28774, 28773, 28771, 28770, 28768, 28767, 28765, 28764, 28762, 28761, 28759, 28758,
28756, 28755, 28753, 28752, 28750, 28749, 28747, 28746, 28744, 28743, 28741, 28740, 28738, 28737, 28735, 28734, 28732, 28731,
28729, 28728, 28726, 28725, 28723, 28722, 28720, 28719, 28717, 28716, 28714, 28713, 28711, 28710, 28708, 28707, 28705, 28704,
28702, 28701, 28699, 28697, 28696, 28694, 28693, 28691, 28690, 28688, 28687, 28685, 28684, 28682, 28681, 28679, 28678, 28676,
28675, 28673, 28672, 28670, 28669, 28667, 28666, 28664, 28663, 28661, 28659, 28658, 28656, 28655, 28653, 28652, 28650, 28649,
28647, 28646, 28644, 28643, 28641, 28640, 28638, 28637, 28635, 28634, 28632, 28630, 28629, 28627, 28626, 28624, 28623, 28621,
28620, 28618, 28617, 28615, 28614, 28612, 28611, 28609, 28608, 28606, 28604, 28603, 28601, 28600, 28598, 28597, 28595, 28594,
28592, 28591, 28589, 28588, 28586, 28585, 28583, 28581, 28580, 28578, 28577, 28575, 28574, 28572, 28571, 28569, 28568, 28566,
28565, 28563, 28561, 28560, 28558, 28557, 28555, 28554, 28552, 28551, 28549, 28548, 28546, 28545, 28543, 28541, 28540, 28538,
28537, 28535, 28534, 28532, 28531, 28529, 28528, 28526, 28524, 28523, 28521, 28520, 28518, 28517, 28515, 28514, 28512, 28511,
28509, 28507, 28506, 28504, 28503, 28501, 28500, 28498, 28497, 28495, 28493, 28492, 28490, 28489, 28487, 28486, 28484, 28483,
28481, 28479, 28478, 28476, 28475, 28473, 28472, 28470, 28469, 28467, 28465, 28464, 28462, 28461, 28459, 28458, 28456, 28455,
28453, 28451, 28450, 28448, 28447, 28445, 28444, 28442, 28441, 28439, 28437, 28436, 28434, 28433, 28431, 28430, 28428, 28427,
28425, 28423, 28422, 28420, 28419, 28417, 28416, 28414, 28412, 28411, 28409, 28408, 28406, 28405, 28403, 28401, 28400, 28398,
28397, 28395, 28394, 28392, 28391, 28389, 28387, 28386, 28384, 28383, 28381, 28380, 28378, 28376, 28375, 28373, 28372, 28370,
28369, 28367, 28365, 28364, 28362, 28361, 28359, 28358, 28356, 28354, 28353, 28351, 28350, 28348, 28346, 28345, 28343, 28342,
28340, 28339, 28337, 28335, 28334, 28332, 28331, 28329, 28328, 28326, 28324, 28323, 28321, 28320, 28318, 28316, 28315, 28313,
28312, 28310, 28309, 28307, 28305, 28304, 28302, 28301, 28299, 28297, 28296, 28294, 28293, 28291, 28290, 28288, 28286, 28285,
28283, 28282, 28280, 28278, 28277, 28275, 28274, 28272, 28271, 28269, 28267, 28266, 28264, 28263, 28261, 28259, 28258, 28256,
28255, 28253, 28251, 28250, 28248, 28247, 28245, 28243, 28242, 28240, 28239, 28237, 28236, 28234, 28232, 28231, 28229, 28228,
28226, 28224, 28223, 28221, 28220, 28218, 28216, 28215, 28213, 28212, 28210, 28208, 28207, 28205, 28204, 28202, 28200, 28199,
28197, 28196, 28194, 28192, 28191, 28189, 28188, 28186, 28184, 28183, 28181, 28180, 28178, 28176, 28175, 28173, 28172, 28170,
28168, 28167, 28165, 28164, 28162, 28160, 28159, 28157, 28155, 28154, 28152, 28151, 28149, 28147, 28146, 28144, 28143, 28141,
28139, 28138, 28136, 28135, 28133, 28131, 28130, 28128, 28127, 28125, 28123, 28122, 28120, 28118, 28117, 28115, 28114, 28112,
28110, 28109, 28107, 28106, 28104, 28102, 28101, 28099, 28097, 28096, 28094, 28093, 28091, 28089, 28088, 28086, 28085, 28083,
28081, 28080, 28078, 28076, 28075, 28073, 28072, 28070, 28068, 28067, 28065, 28063, 28062, 28060, 28059, 28057, 28055, 28054,
28052, 28050, 28049, 28047, 28046, 28044, 28042, 28041, 28039, 28037, 28036, 28034, 28033, 28031, 28029, 28028, 28026, 28024,
28023, 28021, 28020, 28018, 28016, 28015, 28013, 28011, 28010, 28008, 28007, 28005, 28003, 28002, 28000, 27998, 27997, 27995,
27993, 27992, 27990, 27989, 27987, 27985, 27984, 27982, 27980, 27979, 27977, 27976, 27974, 27972, 27971, 27969, 27967, 27966,
27964, 27962, 27961, 27959, 27958, 27956, 27954, 27953, 27951, 27949, 27948, 27946, 27944, 27943, 27941, 27939, 27938, 27936,
27935, 27933, 27931, 27930, 27928, 27926, 27925, 27923, 27921, 27920, 27918, 27916, 27915, 27913, 27912, 27910, 27908, 27907,
27905, 27903, 27902, 27900, 27898, 27897, 27895, 27893, 27892, 27890, 27888, 27887, 27885, 27884, 27882, 27880, 27879, 27877,
27875, 27874, 27872, 27870, 27869, 27867, 27865, 27864, 27862, 27860, 27859, 27857, 27855, 27854, 27852, 27850, 27849, 27847,
27845, 27844, 27842, 27841, 27839, 27837, 27836, 27834, 27832, 27831, 27829, 27827, 27826, 27824, 27822, 27821, 27819, 27817,
27816, 27814, 27812, 27811, 27809, 27807, 27806, 27804, 27802, 27801, 27799, 27797, 27796, 27794, 27792, 27791, 27789, 27787,

27786, 27784, 27782, 27781, 27779, 27777, 27776, 27774, 27772, 27771, 27769, 27767, 27766, 27764, 27762, 27761, 27759, 27757,
27756, 27754, 27752, 27751, 27749, 27747, 27746, 27744, 27742, 27741, 27739, 27737, 27736, 27734, 27732, 27731, 27729, 27727,
27726, 27724, 27722, 27721, 27719, 27717, 27716, 27714, 27712, 27711, 27709, 27707, 27705, 27704, 27702, 27700, 27699, 27697,
27695, 27694, 27692, 27690, 27689, 27687, 27685, 27684, 27682, 27680, 27679, 27677, 27675, 27674, 27672, 27670, 27669, 27667,
27665, 27663, 27662, 27660, 27658, 27657, 27655, 27653, 27652, 27650, 27648, 27647, 27645, 27643, 27642, 27640, 27638, 27637,
27635, 27633, 27631, 27630, 27628, 27626, 27625, 27623, 27621, 27620, 27618, 27616, 27615, 27613, 27611, 27609, 27608, 27606,
27604, 27603, 27601, 27599, 27598, 27596, 27594, 27593, 27591, 27589, 27587, 27586, 27584, 27582, 27581, 27579, 27577, 27576,
27574, 27572, 27570, 27569, 27567, 27565, 27564, 27562, 27560, 27559, 27557, 27555, 27553, 27552, 27550, 27548, 27547, 27545,
27543, 27542, 27540, 27538, 27536, 27535, 27533, 27531, 27530, 27528, 27526, 27525, 27523, 27521, 27519, 27518, 27516, 27514,
27513, 27511, 27509, 27507, 27506, 27504, 27502, 27501, 27499, 27497, 27496, 27494, 27492, 27490, 27489, 27487, 27485, 27484,
27482, 27480, 27478, 27477, 27475, 27473, 27472, 27470, 27468, 27466, 27465, 27463, 27461, 27460, 27458, 27456, 27454, 27453,
27451, 27449, 27448, 27446, 27444, 27442, 27441, 27439, 27437, 27436, 27434, 27432, 27430, 27429, 27427, 27425, 27424, 27422,
27420, 27418, 27417, 27415, 27413, 27411, 27410, 27408, 27406, 27405, 27403, 27401, 27399, 27398, 27396, 27394, 27393, 27391,
27389, 27387, 27386, 27384, 27382, 27380, 27379, 27377, 27375, 27374, 27372, 27370, 27368, 27367, 27365, 27363, 27361, 27360,
27358, 27356, 27355, 27353, 27351, 27349, 27348, 27346, 27344, 27342, 27341, 27339, 27337, 27336, 27334, 27332, 27330, 27329,
27327, 27325, 27323, 27322, 27320, 27318, 27316, 27315, 27313, 27311, 27310, 27308, 27306, 27304, 27303, 27301, 27299, 27297,
27296, 27294, 27292, 27290, 27289, 27287, 27285, 27283, 27282, 27280, 27278, 27276, 27275, 27273, 27271, 27270, 27268, 27266,
27264, 27263, 27261, 27259, 27257, 27256, 27254, 27252, 27250, 27249, 27247, 27245, 27243, 27242, 27240, 27238, 27236, 27235,
27233, 27231, 27229, 27228, 27226, 27224, 27222, 27221, 27219, 27217, 27215, 27214, 27212, 27210, 27208, 27207, 27205, 27203,
27201, 27200, 27198, 27196, 27194, 27193, 27191, 27189, 27187, 27186, 27184, 27182, 27180, 27179, 27177, 27175, 27173, 27172,
27170, 27168, 27166, 27165, 27163, 27161, 27159, 27158, 27156, 27154, 27152, 27150, 27149, 27147, 27145, 27143, 27142, 27140,
27138, 27136, 27135, 27133, 27131, 27129, 27128, 27126, 27124, 27122, 27121, 27119, 27117, 27115, 27113, 27112, 27110, 27108,
27106, 27105, 27103, 27101, 27099, 27098, 27096, 27094, 27092, 27091, 27089, 27087, 27085, 27083, 27082, 27080, 27078, 27076,
27075, 27073, 27071, 27069, 27068, 27066, 27064, 27062, 27060, 27059, 27057, 27055, 27053, 27052, 27050, 27048, 27046, 27045,
27043, 27041, 27039, 27037, 27036, 27034, 27032, 27030, 27029, 27027, 27025, 27023, 27021, 27020, 27018, 27016, 27014, 27013,
27011, 27009, 27007, 27005, 27004, 27002, 27000, 26998, 26997, 26995, 26993, 26991, 26989, 26988, 26986, 26984, 26982, 26980,
26979, 26977, 26975, 26973, 26972, 26970, 26968, 26966, 26964, 26963, 26961, 26959, 26957, 26955, 26954, 26952, 26950, 26948,
26947, 26945, 26943, 26941, 26939, 26938, 26936, 26934, 26932, 26930, 26929, 26927, 26925, 26923, 26922, 26920, 26918, 26916,
26914, 26913, 26911, 26909, 26907, 26905, 26904, 26902, 26900, 26898, 26896, 26895, 26893, 26891, 26889, 26887, 26886, 26884,
26882, 26880, 26878, 26877, 26875, 26873, 26871, 26869, 26868, 26866, 26864, 26862, 26860, 26859, 26857, 26855, 26853, 26851,
26850, 26848, 26846, 26844, 26842, 26841, 26839, 26837, 26835, 26833, 26832, 26830, 26828, 26826, 26824, 26823, 26821, 26819,
26817, 26815, 26814, 26812, 26810, 26808, 26806, 26805, 26803, 26801, 26799, 26797, 26796, 26794, 26792, 26790, 26788, 26787,
26785, 26783, 26781, 26779, 26777, 26776, 26774, 26772, 26770, 26768, 26767, 26765, 26763, 26761, 26759, 26758, 26756, 26754,
26752, 26750, 26748, 26747, 26745, 26743, 26741, 26739, 26738, 26736, 26734, 26732, 26730, 26728, 26727, 26725, 26723, 26721,
26719, 26718, 26716, 26714, 26712, 26710, 26708, 26707, 26705, 26703, 26701, 26699, 26698, 26696, 26694, 26692, 26690, 26688,
26687, 26685, 26683, 26681, 26679, 26677, 26676, 26674, 26672, 26670, 26668, 26667, 26665, 26663, 26661, 26659, 26657, 26656,
26654, 26652, 26650, 26648, 26646, 26645, 26643, 26641, 26639, 26637, 26635, 26634, 26632, 26630, 26628, 26626, 26624, 26623,
26621, 26619, 26617, 26615, 26613, 26612, 26610, 26608, 26606, 26604, 26602, 26601, 26599, 26597, 26595, 26593, 26591, 26590,
26588, 26586, 26584, 26582, 26580, 26579, 26577, 26575, 26573, 26571, 26569, 26568, 26566, 26564, 26562, 26560, 26558, 26557,
26555, 26553, 26551, 26549, 26547, 26546, 26544, 26542, 26540, 26538, 26536, 26534, 26533, 26531, 26529, 26527, 26525, 26523,
26522, 26520, 26518, 26516, 26514, 26512, 26510, 26509, 26507, 26505, 26503, 26501, 26499, 26498, 26496, 26494, 26492, 26490,
26488, 26486, 26485, 26483, 26481, 26479, 26477, 26475, 26473, 26472, 26470, 26468, 26466, 26464, 26462, 26461, 26459, 26457,
26455, 26453, 26451, 26449, 26448, 26446, 26444, 26442, 26440, 26438, 26436, 26435, 26433, 26431, 26429, 26427, 26425, 26423,
26422, 26420, 26418, 26416, 26414, 26412, 26410, 26409, 26407, 26405, 26403, 26401, 26399, 26397, 26396, 26394, 26392, 26390,
26388, 26386, 26384, 26382, 26381, 26379, 26377, 26375, 26373, 26371, 26369, 26368, 26366, 26364, 26362, 26360, 26358, 26356,
26355, 26353, 26351, 26349, 26347, 26345, 26343, 26341, 26340, 26338, 26336, 26334, 26332, 26330, 26328, 26326, 26325, 26323,
26321, 26319, 26317, 26315, 26313, 26312, 26310, 26308, 26306, 26304, 26302, 26300, 26298, 26297, 26295, 26293, 26291, 26289,
26287, 26285, 26283, 26282, 26280, 26278, 26276, 26274, 26272, 26270, 26268, 26267, 26265, 26263, 26261, 26259, 26257, 26255,
26253, 26251, 26250, 26248, 26246, 26244, 26242, 26240, 26238, 26236, 26235, 26233, 26231, 26229, 26227, 26225, 26223, 26221,
26219, 26218, 26216, 26214, 26212, 26210, 26208, 26206, 26204, 26203, 26201, 26199, 26197, 26195, 26193, 26191, 26189, 26187,
26186, 26184, 26182, 26180, 26178, 26176, 26174, 26172, 26170, 26169, 26167, 26165, 26163, 26161, 26159, 26157, 26155, 26153,
26151, 26150, 26148, 26146, 26144, 26142, 26140, 26138, 26136, 26134, 26133, 26131, 26129, 26127, 26125, 26123, 26121, 26119,
26117, 26115, 26114, 26112, 26110, 26108, 26106, 26104, 26102, 26100, 26098, 26096, 26095, 26093, 26091, 26089, 26087, 26085,
26083, 26081, 26079, 26077, 26076, 26074, 26072, 26070, 26068, 26066, 26064, 26062, 26060, 26058, 26057, 26055, 26053, 26051,
26049, 26047, 26045, 26043, 26041, 26039, 26037, 26036, 26034, 26032, 26030, 26028, 26026, 26024, 26022, 26020, 26018, 26016,
26015, 26013, 26011, 26009, 26007, 26005, 26003, 26001, 25999, 25997, 25995, 25994, 25992, 25990, 25988, 25986, 25984, 25982,
25980, 25978, 25976, 25974, 25972, 25971, 25969, 25967, 25965, 25963, 25961, 25959, 25957, 25955, 25953, 25951, 25949, 25948,
25946, 25944, 25942, 25940, 25938, 25936, 25934, 25932, 25930, 25928, 25926, 25925, 25923, 25921, 25919, 25917, 25915, 25913,
25911, 25909, 25907, 25905, 25903, 25901, 25900, 25898, 25896, 25894, 25892, 25890, 25888, 25886, 25884, 25882, 25880, 25878,
25876, 25875, 25873, 25871, 25869, 25867, 25865, 25863, 25861, 25859, 25857, 25855, 25853, 25851, 25849, 25847, 25846, 25844,
25842, 25840, 25838, 25836, 25834, 25832, 25830, 25828, 25826, 25824, 25822, 25820, 25819, 25817, 25815, 25813, 25811, 25809,
25807, 25805, 25803, 25801, 25799, 25797, 25795, 25793, 25791, 25789, 25788, 25786, 25784, 25782, 25780, 25778, 25776, 25774,
25772, 25770, 25768, 25766, 25764, 25762, 25760, 25758, 25756, 25755, 25753, 25751, 25749, 25747, 25745, 25743, 25741, 25739,
25737, 25735, 25733, 25731, 25729, 25727, 25725, 25723, 25721, 25720, 25718, 25716, 25714, 25712, 25710, 25708, 25706, 25704,
25702, 25700, 25698, 25696, 25694, 25692, 25690, 25688, 25686, 25684, 25683, 25681, 25679, 25677, 25675, 25673, 25671, 25669,
25667, 25665, 25663, 25661, 25659, 25657, 25655, 25653, 25651, 25649, 25647, 25645, 25643, 25641, 25640, 25638, 25636, 25634,
25632, 25630, 25628, 25626, 25624, 25622, 25620, 25618, 25616, 25614, 25612, 25610, 25608, 25606, 25604, 25602, 25600, 25598,
25596, 25594, 25593, 25591, 25589, 25587, 25585, 25583, 25581, 25579, 25577, 25575, 25573, 25571, 25569, 25567, 25565, 25563,
25561, 25559, 25557, 25555, 25553, 25551, 25549, 25547, 25545, 25543, 25541, 25539, 25537, 25536, 25534, 25532, 25530, 25528,
25526, 25524, 25522, 25520, 25518, 25516, 25514, 25512, 25510, 25508, 25506, 25504, 25502, 25500, 25498, 25496, 25494, 25492,
25490, 25488, 25486, 25484, 25482, 25480, 25478, 25476, 25474, 25472, 25470, 25468, 25466, 25464, 25463, 25461, 25459, 25457,

25455, 25453, 25451, 25449, 25447, 25445, 25443, 25441, 25439, 25437, 25435, 25433, 25431, 25429, 25427, 25425, 25423, 25421, 25419, 25417, 25415, 25413, 25411, 25409, 25407, 25405, 25403, 25401, 25399, 25397, 25395, 25393, 25391, 25389, 25387, 25385, 25383, 25381, 25379, 25377, 25375, 25373, 25371, 25369, 25367, 25365, 25363, 25361, 25359, 25357, 25355, 25353, 25351, 25349, 25347, 25345, 25343, 25341, 25339, 25337, 25335, 25333, 25331, 25330, 25328, 25326, 25324, 25322, 25320, 25318, 25316, 25314, 25312, 25310, 25308, 25306, 25304, 25302, 25300, 25298, 25296, 25294, 25292, 25290, 25288, 25286, 25284, 25282, 25280, 25278, 25276, 25274, 25272, 25270, 25268, 25266, 25264, 25262, 25260, 25258, 25256, 25254, 25252, 25250, 25248, 25246, 25244, 25242, 25240, 25238, 25236, 25234, 25232, 25230, 25228, 25226, 25224, 25222, 25220, 25218, 25216, 25214, 25212, 25210, 25208, 25205, 25203, 25201, 25199, 25197, 25195, 25193, 25191, 25189, 25187, 25185, 25183, 25181, 25179, 25177, 25175, 25173, 25171, 25169, 25167, 25165, 25163, 25161, 25159, 25157, 25155, 25153, 25151, 25149, 25147, 25145, 25143, 25141, 25139, 25137, 25135, 25133, 25131, 25129, 25127, 25125, 25123, 25121, 25119, 25117, 25115, 25113, 25111, 25109, 25107, 25105, 25103, 25101, 25099, 25097, 25095, 25093, 25091, 25089, 25087, 25085, 25083, 25081, 25079, 25077, 25075, 25073, 25070, 25068, 25066, 25064, 25062, 25060, 25058, 25056, 25054, 25052, 25050, 25048, 25046, 25044, 25042, 25040, 25038, 25036, 25034, 25032, 25030, 25028, 25026, 25024, 25022, 25020, 25018, 25016, 25014, 25012, 25010, 25008, 25006, 25004, 25002, 25000, 24998, 24995, 24993, 24991, 24989, 24987, 24985, 24983, 24981, 24979, 24977, 24975, 24973, 24971, 24969, 24967, 24965, 24963, 24961, 24959, 24957, 24955, 24953, 24951, 24949, 24947, 24945, 24943, 24941, 24939, 24936, 24934, 24932, 24930, 24928, 24926, 24924, 24922, 24920, 24918, 24916, 24914, 24912, 24910, 24908, 24906, 24904, 24902, 24900, 24898, 24896, 24894, 24892, 24890, 24887, 24885, 24883, 24881, 24879, 24877, 24875, 24873, 24871, 24869, 24867, 24865, 24863, 24861, 24859, 24857, 24855, 24853, 24851, 24849, 24847, 24845, 24842, 24840, 24838, 24836, 24834, 24832, 24830, 24828, 24826, 24824, 24822, 24820, 24818, 24816, 24814, 24812, 24810, 24808, 24806, 24804, 24801, 24799, 24797, 24795, 24793, 24791, 24789, 24787, 24785, 24783, 24781, 24779, 24777, 24775, 24773, 24771, 24769, 24767, 24764, 24762, 24760, 24758, 24756, 24754, 24752, 24750, 24748, 24746, 24744, 24742, 24740, 24738, 24736, 24734, 24732, 24729, 24727, 24725, 24723, 24721, 24719, 24717, 24715, 24713, 24711, 24709, 24707, 24705, 24703, 24701, 24699, 24696, 24694, 24692, 24690, 24688, 24686, 24684, 24682, 24680, 24678, 24676, 24674, 24672, 24670, 24668, 24665, 24663, 24661, 24659, 24657, 24655, 24653, 24651, 24649, 24647, 24645, 24643, 24641, 24639, 24636, 24634, 24632, 24630, 24628, 24626, 24624, 24622, 24620, 24618, 24616, 24614, 24612, 24610, 24607, 24605, 24603, 24601, 24599, 24597, 24595, 24593, 24591, 24589, 24587, 24585, 24583, 24580, 24578, 24576, 24574, 24572, 24570, 24568, 24566, 24564, 24562, 24560, 24558, 24556, 24553, 24551, 24549, 24547, 24545, 24543, 24541, 24539, 24537, 24535, 24533, 24531, 24528, 24526, 24524, 24522, 24520, 24518, 24516, 24514, 24512, 24510, 24508, 24506, 24503, 24501, 24499, 24497, 24495, 24493, 24491, 24489, 24487, 24485, 24483, 24480, 24478, 24476, 24474, 24472, 24470, 24468, 24466, 24464, 24462, 24460, 24458, 24455, 24453, 24451, 24449, 24447, 24445, 24443, 24441, 24439, 24437, 24434, 24432, 24430, 24428, 24426, 24424, 24422, 24420, 24418, 24416, 24414, 24411, 24409, 24407, 24405, 24403, 24401, 24399, 24397, 24395, 24393, 24390, 24388, 24386, 24384, 24382, 24380, 24378, 24376, 24374, 24372, 24369, 24367, 24365, 24363, 24361, 24359, 24357, 24355, 24353, 24351, 24348, 24346, 24344, 24342, 24340, 24338, 24336, 24334, 24332, 24330, 24327, 24325, 24323, 24321, 24319, 24317, 24315, 24313, 24311, 24309, 24306, 24304, 24302, 24300, 24298, 24296, 24294, 24292, 24290, 24287, 24285, 24283, 24281, 24279, 24277, 24275, 24273, 24271, 24268, 24266, 24264, 24262, 24260, 24258, 24256, 24254, 24252, 24249, 24247, 24245, 24243, 24241, 24239, 24237, 24235, 24233, 24230, 24228, 24226, 24224, 24222, 24220, 24218, 24216, 24213, 24211, 24209, 24207, 24205, 24203, 24201, 24199, 24197, 24194, 24192, 24190, 24188, 24186, 24184, 24182, 24180, 24177, 24175, 24173, 24171, 24169, 24167, 24165, 24163, 24160, 24158, 24156, 24154, 24152, 24150, 24148, 24146, 24144, 24141, 24139, 24137, 24135, 24133, 24131, 24129, 24127, 24124, 24122, 24120, 24118, 24116, 24114, 24112, 24109, 24107, 24105, 24103, 24101, 24099, 24097, 24095, 24092, 24090, 24088, 24086, 24084, 24082, 24080, 24078, 24075, 24073, 24071, 24069, 24067, 24065, 24063, 24061, 24058, 24056, 24054, 24052, 24050, 24048, 24046, 24043, 24041, 24039, 24037, 24035, 24033, 24031, 24028, 24026, 24024, 24022, 24020, 24018, 24016, 24014, 24011, 24009, 24007, 24005, 24003, 24001, 23999, 23996, 23994, 23992, 23990, 23988, 23986, 23984, 23981, 23979, 23977, 23975, 23973, 23971, 23969, 23966, 23964, 23962, 23960, 23958, 23956, 23954, 23951, 23949, 23947, 23945, 23943, 23941, 23939, 23936, 23934, 23932, 23930, 23928, 23926, 23924, 23921, 23919, 23917, 23915, 23913, 23911, 23909, 23906, 23904, 23902, 23900, 23898, 23896, 23893, 23891, 23889, 23887, 23885, 23883, 23881, 23878, 23876, 23874, 23872, 23870, 23868, 23866, 23863, 23861, 23859, 23857, 23855, 23853, 23850, 23848, 23846, 23844, 23842, 23840, 23838, 23835, 23833, 23831, 23829, 23827, 23825, 23822, 23820, 23818, 23816, 23814, 23812, 23809, 23807, 23805, 23803, 23801, 23799, 23797, 23794, 23792, 23790, 23788, 23786, 23784, 23781, 23779, 23777, 23775, 23773, 23771, 23768, 23766, 23764, 23762, 23760, 23758, 23755, 23753, 23751, 23749, 23747, 23745, 23742, 23740, 23738, 23736, 23734, 23732, 23729, 23727, 23725, 23723, 23721, 23719, 23716, 23714, 23712, 23710, 23708, 23706, 23703, 23701, 23699, 23697, 23695, 23693, 23690, 23688, 23686, 23684, 23682, 23680, 23677, 23675, 23673, 23671, 23669, 23667, 23664, 23662, 23660, 23658, 23656, 23654, 23651, 23649, 23647, 23645, 23643, 23640, 23638, 23636, 23634, 23632, 23630, 23627, 23625, 23623, 23621, 23619, 23617, 23614, 23612, 23610, 23608, 23606, 23603, 23601, 23599, 23597, 23595, 23593, 23590, 23588, 23586, 23584, 23582, 23579, 23577, 23575, 23573, 23571, 23569, 23566, 23564, 23562, 23560, 23558, 23555, 23553, 23551, 23549, 23547, 23545, 23542, 23540, 23538, 23536, 23534, 23531, 23529, 23527, 23525, 23523, 23520, 23518, 23516, 23514, 23512, 23510, 23507, 23505, 23503, 23501, 23499, 23496, 23494, 23492, 23490, 23488, 23485, 23483, 23481, 23479, 23477, 23474, 23472, 23470, 23468, 23466, 23464, 23461, 23459, 23457, 23455, 23453, 23450, 23448, 23446, 23444, 23442, 23439, 23437, 23435, 23433, 23431, 23428, 23426, 23424, 23422, 23420, 23417, 23415, 23413, 23411, 23409, 23406, 23404, 23402, 23400, 23398, 23395, 23393, 23391, 23389, 23387, 23384, 23382, 23380, 23378, 23376, 23373, 23371, 23369, 23367, 23365, 23362, 23360, 23358, 23356, 23354, 23351, 23349, 23347, 23345, 23343, 23340, 23338, 23336, 23334, 23332, 23329, 23327, 23325, 23323, 23321, 23318, 23316, 23314, 23312, 23310, 23307, 23305, 23303, 23301, 23298, 23296, 23294, 23292, 23290, 23287, 23285, 23283, 23281, 23279, 23276, 23274, 23272, 23270, 23268, 23265, 23263, 23261, 23259, 23256, 23254, 23252, 23250, 23248, 23245, 23243, 23241, 23239, 23237, 23234, 23232, 23230, 23228, 23225, 23223, 23221, 23219, 23217, 23214, 23212, 23210, 23208, 23205, 23203, 23201, 23199, 23197, 23194, 23192, 23190, 23188, 23186, 23183, 23181, 23179, 23177, 23174, 23172, 23170, 23168, 23166, 23163, 23161, 23159, 23157, 23154, 23152, 23150, 23148, 23146, 23143, 23141, 23139, 23137, 23134, 23132, 23130, 23128, 23126, 23123, 23121, 23119, 23117, 23114, 23112, 23110, 23108, 23105, 23103, 23101, 23099, 23097, 23094, 23092, 23090, 23088, 23085, 23083, 23081, 23079, 23076, 23074, 23072, 23070, 23068, 23065, 23063, 23061, 23059, 23056, 23054, 23052, 23050, 23047, 23045, 23043, 23041, 23039, 23036, 23034, 23032, 23030, 23027, 23025, 23023, 23021, 23018, 23016, 23014, 23012, 23009, 23007, 23005, 23003, 23001, 22998, 22996, 22994, 22992, 22989, 22987, 22985, 22983, 22980, 22978, 22976, 22974, 22971, 22969, 22967, 22965, 22962, 22960, 22958, 22956, 22953, 22951, 22949, 22947, 22945, 22942, 22940, 22938, 22936, 22933, 22931, 22929, 22927, 22924, 22922, 22920, 22918, 22915, 22913, 22911, 22909, 22906, 22904, 22902, 22900, 22897, 22895, 22893, 22891, 22888, 22886, 22884, 22882, 22879, 22877, 22875, 22873, 22870, 22868, 22866, 22864, 22861, 22859, 22857, 22855, 22852, 22850, 22848, 22846, 22843, 22841, 22839, 22837, 22834, 22832, 22830, 22828, 22825, 22823, 22821, 22819, 22816, 22814, 22812, 22810, 22807, 22805, 22803, 22801, 22798, 22796, 22794, 22792, 22789, 22787, 22785, 22782, 22780, 22778, 22776, 22773, 22771, 22769, 22767, 22764, 22762, 22760, 22758, 22755, 22753, 22751, 22749, 22746, 22744,

22742, 22740, 22737, 22735, 22733, 22731, 22728, 22726, 22724, 22721, 22719, 22717, 22715, 22712, 22710, 22708, 22706, 22703, 22701, 22699, 22697, 22694, 22692, 22690, 22687, 22685, 22683, 22681, 22678, 22676, 22674, 22672, 22669, 22667, 22665, 22663, 22660, 22658, 22656, 22653, 22651, 22649, 22647, 22644, 22642, 22640, 22638, 22635, 22633, 22631, 22628, 22626, 22624, 22622, 22619, 22617, 22615, 22613, 22610, 22608, 22606, 22603, 22601, 22599, 22597, 22594, 22592, 22590, 22588, 22585, 22583, 22581, 22578, 22576, 22574, 22572, 22569, 22567, 22565, 22562, 22560, 22558, 22556, 22553, 22551, 22549, 22547, 22544, 22542, 22540, 22537, 22535, 22533, 22531, 22528, 22526, 22524, 22521, 22519, 22517, 22515, 22512, 22510, 22508, 22505, 22503, 22501, 22499, 22496, 22494, 22492, 22489, 22487, 22485, 22483, 22480, 22478, 22476, 22473, 22471, 22469, 22467, 22464, 22462, 22460, 22457, 22455, 22453, 22451, 22448, 22446, 22444, 22441, 22439, 22437, 22435, 22432, 22430, 22428, 22425, 22423, 22421, 22419, 22416, 22414, 22412, 22409, 22407, 22405, 22403, 22400, 22398, 22396, 22393, 22391, 22389, 22386, 22384, 22382, 22380, 22377, 22375, 22373, 22370, 22368, 22366, 22364, 22361, 22359, 22357, 22354, 22352, 22350, 22347, 22345, 22343, 22341, 22338, 22336, 22334, 22331, 22329, 22327, 22324, 22322, 22320, 22318, 22315, 22313, 22311, 22308, 22306, 22304, 22301, 22299, 22297, 22295, 22292, 22290, 22288, 22285, 22283, 22281, 22278, 22276, 22274, 22271, 22269, 22267, 22265, 22262, 22260, 22258, 22255, 22253, 22251, 22248, 22246, 22244, 22242, 22239, 22237, 22235, 22232, 22230, 22228, 22225, 22223, 22221, 22218, 22216, 22214, 22212, 22209, 22207, 22205, 22202, 22200, 22198, 22195, 22193, 22191, 22188, 22186, 22184, 22181, 22179, 22177, 22175, 22172, 22170, 22168, 22165, 22163, 22161, 22158, 22156, 22154, 22151, 22149, 22147, 22144, 22142, 22140, 22137, 22135, 22133, 22131, 22128, 22126, 22124, 22121, 22119, 22117, 22114, 22112, 22110, 22107, 22105, 22103, 22100, 22098, 22096, 22093, 22091, 22089, 22086, 22084, 22082, 22080, 22077, 22075, 22073, 22070, 22068, 22066, 22063, 22061, 22059, 22056, 22054, 22052, 22049, 22047, 22045, 22042, 22040, 22038, 22035, 22033, 22031, 22028, 22026, 22024, 22021, 22019, 22017, 22014, 22012, 22010, 22007, 22005, 22003, 22000, 21998, 21996, 21994, 21991, 21989, 21987, 21984, 21982, 21980, 21977, 21975, 21973, 21970, 21968, 21966, 21963, 21961, 21959, 21956, 21954, 21952, 21949, 21947, 21945, 21942, 21940, 21938, 21935, 21933, 21931, 21928, 21926, 21924, 21921, 21919, 21917, 21914, 21912, 21910, 21907, 21905, 21903, 21900, 21898, 21896, 21893, 21891, 21889, 21886, 21884, 21881, 21879, 21877, 21874, 21872, 21870, 21867, 21865, 21863, 21860, 21858, 21856, 21853, 21851, 21849, 21846, 21844, 21842, 21839, 21837, 21835, 21832, 21830, 21828, 21825, 21823, 21821, 21818, 21816, 21814, 21811, 21809, 21807, 21804, 21802, 21800, 21797, 21795, 21792, 21790, 21788, 21785, 21783, 21781, 21778, 21776, 21774, 21771, 21769, 21767, 21764, 21762, 21760, 21757, 21755, 21753, 21750, 21748, 21746, 21743, 21741, 21738, 21736, 21734, 21731, 21729, 21727, 21724, 21722, 21720, 21717, 21715, 21713, 21710, 21708, 21706, 21703, 21701, 21698, 21696, 21694, 21691, 21689, 21687, 21684, 21682, 21680, 21677, 21675, 21673, 21670, 21668, 21666, 21663, 21661, 21658, 21656, 21654, 21651, 21649, 21647, 21644, 21642, 21640, 21637, 21635, 21632, 21630, 21628, 21625, 21623, 21621, 21618, 21616, 21614, 21611, 21609, 21607, 21604, 21602, 21599, 21597, 21595, 21592, 21590, 21588, 21585, 21583, 21581, 21578, 21576, 21573, 21571, 21569, 21566, 21564, 21562, 21559, 21557, 21555, 21552, 21550, 21547, 21545, 21543, 21540, 21538, 21536, 21533, 21531, 21528, 21526, 21524, 21521, 21519, 21517, 21514, 21512, 21510, 21507, 21505, 21502, 21500, 21498, 21495, 21493, 21491, 21488, 21486, 21483, 21481, 21479, 21476, 21474, 21472, 21469, 21467, 21464, 21462, 21460, 21457, 21455, 21453, 21450, 21448, 21445, 21443, 21441, 21438, 21436, 21434, 21431, 21429, 21426, 21424, 21422, 21419, 21417, 21415, 21412, 21410, 21407, 21405, 21403, 21400, 21398, 21396, 21393, 21391, 21388, 21386, 21384, 21381, 21379, 21376, 21374, 21372, 21369, 21367, 21365, 21362, 21360, 21357, 21355, 21353, 21350, 21348, 21346, 21343, 21341, 21338, 21336, 21334, 21331, 21329, 21326, 21324, 21322, 21319, 21317, 21315, 21312, 21310, 21307, 21305, 21303, 21300, 21298, 21295, 21293, 21291, 21288, 21286, 21283, 21281, 21279, 21276, 21274, 21272, 21269, 21267, 21264, 21262, 21260, 21257, 21255, 21252, 21250, 21248, 21245, 21243, 21240, 21238, 21236, 21233, 21231, 21228, 21226, 21224, 21221, 21219, 21217, 21214, 21212, 21209, 21207, 21205, 21202, 21200, 21197, 21195, 21193, 21190, 21188, 21185, 21183, 21181, 21178, 21176, 21173, 21171, 21169, 21166, 21164, 21161, 21159, 21157, 21154, 21152, 21149, 21147, 21145, 21142, 21140, 21137, 21135, 21133, 21130, 21128, 21125, 21123, 21121, 21118, 21116, 21113, 21111, 21109, 21106, 21104, 21101, 21099, 21097, 21094, 21092, 21089, 21087, 21085, 21082, 21080, 21077, 21075, 21073, 21070, 21068, 21065, 21063, 21060, 21058, 21056, 21053, 21051, 21048, 21046, 21044, 21041, 21039, 21036, 21034, 21032, 21029, 21027, 21024, 21022, 21020, 21017, 21015, 21012, 21010, 21007, 21005, 21003, 21000, 20998, 20995, 20993, 20991, 20988, 20986, 20983, 20981, 20979, 20976, 20974, 20971, 20969, 20966, 20964, 20962, 20959, 20957, 20954, 20952, 20950, 20947, 20945, 20942, 20940, 20937, 20935, 20933, 20930, 20928, 20925, 20923, 20921, 20918, 20916, 20913, 20911, 20908, 20906, 20904, 20901, 20899, 20896, 20894, 20892, 20889, 20887, 20884, 20882, 20879, 20877, 20875, 20872, 20870, 20867, 20865, 20862, 20860, 20858, 20855, 20853, 20850, 20848, 20846, 20843, 20841, 20838, 20836, 20833, 20831, 20829, 20826, 20824, 20821, 20819, 20816, 20814, 20812, 20809, 20807, 20804, 20802, 20799, 20797, 20795, 20792, 20790, 20787, 20785, 20782, 20780, 20778, 20775, 20773, 20770, 20768, 20765, 20763, 20761, 20758, 20756, 20753, 20751, 20748, 20746, 20744, 20741, 20739, 20736, 20734, 20731, 20729, 20727, 20724, 20722, 20719, 20717, 20714, 20712, 20709, 20707, 20705, 20702, 20700, 20697, 20695, 20692, 20690, 20688, 20685, 20683, 20680, 20678, 20675, 20673, 20671, 20668, 20666, 20663, 20661, 20658, 20656, 20653, 20651, 20649, 20646, 20644, 20641, 20639, 20636, 20634, 20631, 20629, 20627, 20624, 20622, 20619, 20617, 20614, 20612, 20610, 20607, 20605, 20602, 20600, 20597, 20595, 20592, 20590, 20588, 20585, 20583, 20580, 20578, 20575, 20573, 20570, 20568, 20566, 20563, 20561, 20558, 20556, 20553, 20551, 20548, 20546, 20543, 20541, 20539, 20536, 20534, 20531, 20529, 20526, 20524, 20521, 20519, 20517, 20514, 20512, 20509, 20507, 20504, 20502, 20499, 20497, 20495, 20492, 20490, 20487, 20485, 20482, 20480, 20477, 20475, 20472, 20470, 20468, 20465, 20463, 20460, 20458, 20455, 20453, 20450, 20448, 20445, 20443, 20441, 20438, 20436, 20433, 20431, 20428, 20426, 20423, 20421, 20418, 20416, 20414, 20411, 20409, 20406, 20404, 20401, 20399, 20396, 20394, 20391, 20389, 20386, 20384, 20382, 20379, 20377, 20374, 20372, 20369, 20367, 20364, 20362, 20359, 20357, 20354, 20352, 20350, 20347, 20345, 20342, 20340, 20337, 20335, 20332, 20330, 20327, 20325, 20322, 20320, 20318, 20315, 20313, 20310, 20308, 20305, 20303, 20300, 20298, 20295, 20293, 20290, 20288, 20285, 20283, 20281, 20278, 20276, 20273, 20271, 20268, 20266, 20263, 20261, 20258, 20256, 20253, 20251, 20248, 20246, 20244, 20241, 20239, 20236, 20234, 20231, 20229, 20226, 20224, 20221, 20219, 20216, 20214, 20211, 20209, 20206, 20204, 20201, 20199, 20197, 20194, 20192, 20189, 20187, 20184, 20182, 20179, 20177, 20174, 20172, 20169, 20167, 20164, 20162, 20159, 20157, 20154, 20152, 20150, 20147, 20145, 20142, 20140, 20137, 20135, 20132, 20130, 20127, 20125, 20122, 20120, 20117, 20115, 20112, 20110, 20107, 20105, 20102, 20100, 20097, 20095, 20092, 20090, 20088, 20085, 20083, 20080, 20078, 20075, 20073, 20070, 20068, 20065, 20063, 20060, 20058, 20055, 20053, 20050, 20048, 20045, 20043, 20040, 20038, 20035, 20033, 20030, 20028, 20025, 20023, 20020, 20018, 20015, 20013, 20010, 20008, 20006, 20003, 20001, 19998, 19996, 19993, 19991, 19988, 19986, 19983, 19981, 19978, 19976, 19973, 19971, 19968, 19966, 19963, 19961, 19958, 19956, 19953, 19951, 19948, 19946, 19943, 19941, 19938, 19936, 19933, 19931, 19928, 19926, 19923, 19921, 19918, 19916, 19913, 19911, 19908, 19906, 19903, 19901, 19898, 19896, 19893, 19891, 19888, 19886, 19883, 19881, 19878, 19876, 19873, 19871, 19868, 19866, 19863, 19861, 19858, 19856, 19853, 19851, 19848, 19846, 19843, 19841, 19838, 19836, 19833, 19831, 19828, 19826, 19823, 19821, 19818, 19816, 19813, 19811, 19808, 19806, 19803, 19801, 19798, 19796, 19793, 19791, 19788, 19786, 19783, 19781, 19778, 19776, 19773, 19771, 19768, 19766, 19763, 19761, 19758, 19756, 19753, 19751, 19748, 19746, 19743, 19741, 19738, 19736, 19733, 19731, 19728, 19726, 19723, 19721, 19718, 19716, 19713, 19711, 19708, 19706, 19703, 19701, 19698, 19696, 19693, 19691,

19688, 19686, 19683, 19681, 19678, 19675, 19673, 19670, 19668, 19665, 19663, 19660, 19658, 19655, 19653, 19650, 19648, 19645,
19643, 19640, 19638, 19635, 19633, 19630, 19628, 19625, 19623, 19620, 19618, 19615, 19613, 19610, 19608, 19605, 19603, 19600,
19598, 19595, 19592, 19590, 19587, 19585, 19582, 19580, 19577, 19575, 19572, 19570, 19567, 19565, 19562, 19560, 19557, 19555,
19552, 19550, 19547, 19545, 19542, 19540, 19537, 19535, 19532, 19529, 19527, 19524, 19522, 19519, 19517, 19514, 19512, 19509,
19507, 19504, 19502, 19499, 19497, 19494, 19492, 19489, 19487, 19484, 19482, 19479, 19476, 19474, 19471, 19469, 19466, 19464,
19461, 19459, 19456, 19454, 19451, 19449, 19446, 19444, 19441, 19439, 19436, 19433, 19431, 19428, 19426, 19423, 19421, 19418,
19416, 19413, 19411, 19408, 19406, 19403, 19401, 19398, 19396, 19393, 19390, 19388, 19385, 19383, 19380, 19378, 19375, 19373,
19370, 19368, 19365, 19363, 19360, 19358, 19355, 19352, 19350, 19347, 19345, 19342, 19340, 19337, 19335, 19332, 19330, 19327,
19325, 19322, 19319, 19317, 19314, 19312, 19309, 19307, 19304, 19302, 19299, 19297, 19294, 19292, 19289, 19286, 19284, 19281,
19279, 19276, 19274, 19271, 19269, 19266, 19264, 19261, 19259, 19256, 19253, 19251, 19248, 19246, 19243, 19241, 19238, 19236,
19233, 19231, 19228, 19225, 19223, 19220, 19218, 19215, 19213, 19210, 19208, 19205, 19203, 19200, 19197, 19195, 19192, 19190,
19187, 19185, 19182, 19180, 19177, 19175, 19172, 19169, 19167, 19164, 19162, 19159, 19157, 19154, 19152, 19149, 19147, 19144,
19141, 19139, 19136, 19134, 19131, 19129, 19126, 19124, 19121, 19118, 19116, 19113, 19111, 19108, 19106, 19103, 19101, 19098,
19095, 19093, 19090, 19088, 19085, 19083, 19080, 19078, 19075, 19073, 19070, 19067, 19065, 19062, 19060, 19057, 19055, 19052,
19050, 19047, 19044, 19042, 19039, 19037, 19034, 19032, 19029, 19026, 19024, 19021, 19019, 19016, 19014, 19011, 19009, 19006,
19003, 19001, 18998, 18996, 18993, 18991, 18988, 18986, 18983, 18980, 18978, 18975, 18973, 18970, 18968, 18965, 18962, 18960,
18957, 18955, 18952, 18950, 18947, 18945, 18942, 18939, 18937, 18934, 18932, 18929, 18927, 18924, 18921, 18919, 18916, 18914,
18911, 18909, 18906, 18904, 18901, 18898, 18896, 18893, 18891, 18888, 18886, 18883, 18880, 18878, 18875, 18873, 18870, 18868,
18865, 18862, 18860, 18857, 18855, 18852, 18850, 18847, 18844, 18842, 18839, 18837, 18834, 18832, 18829, 18826, 18824, 18821,
18819, 18816, 18814, 18811, 18808, 18806, 18803, 18801, 18798, 18796, 18793, 18790, 18788, 18785, 18783, 18780, 18778, 18775,
18772, 18770, 18767, 18765, 18762, 18760, 18757, 18754, 18752, 18749, 18747, 18744, 18742, 18739, 18736, 18734, 18731, 18729,
18726, 18723, 18721, 18718, 18716, 18713, 18711, 18708, 18705, 18703, 18700, 18698, 18695, 18693, 18690, 18687, 18685, 18682,
18680, 18677, 18674, 18672, 18669, 18667, 18664, 18662, 18659, 18656, 18654, 18651, 18649, 18646, 18643, 18641, 18638, 18636,
18633, 18631, 18628, 18625, 18623, 18620, 18618, 18615, 18612, 18610, 18607, 18605, 18602, 18600, 18597, 18594, 18592, 18589,
18587, 18584, 18581, 18579, 18576, 18574, 18571, 18568, 18566, 18563, 18561, 18558, 18556, 18553, 18550, 18548, 18545, 18543,
18540, 18537, 18535, 18532, 18530, 18527, 18524, 18522, 18519, 18517, 18514, 18511, 18509, 18506, 18504, 18501, 18499, 18496,
18493, 18491, 18488, 18486, 18483, 18480, 18478, 18475, 18473, 18470, 18467, 18465, 18462, 18460, 18457, 18454, 18452, 18449,
18447, 18444, 18441, 18439, 18436, 18434, 18431, 18428, 18426, 18423, 18421, 18418, 18415, 18413, 18410, 18408, 18405, 18402,
18400, 18397, 18395, 18392, 18389, 18387, 18384, 18382, 18379, 18376, 18374, 18371, 18369, 18366, 18363, 18361, 18358, 18356,
18353, 18350, 18348, 18345, 18343, 18340, 18337, 18335, 18332, 18330, 18327, 18324, 18322, 18319, 18317, 18314, 18311, 18309,
18306, 18304, 18301, 18298, 18296, 18293, 18291, 18288, 18285, 18283, 18280, 18277, 18275, 18272, 18270, 18267, 18264, 18262,
18259, 18257, 18254, 18251, 18249, 18246, 18244, 18241, 18238, 18236, 18233, 18231, 18228, 18225, 18223, 18220, 18217, 18215,
18212, 18210, 18207, 18204, 18202, 18199, 18197, 18194, 18191, 18189, 18186, 18184, 18181, 18178, 18176, 18173, 18170, 18168,
18165, 18163, 18160, 18157, 18155, 18152, 18150, 18147, 18144, 18142, 18139, 18136, 18134, 18131, 18129, 18126, 18123, 18121,
18118, 18116, 18113, 18110, 18108, 18105, 18102, 18100, 18097, 18095, 18092, 18089, 18087, 18084, 18081, 18079, 18076, 18074,
18071, 18068, 18066, 18063, 18061, 18058, 18055, 18053, 18050, 18047, 18045, 18042, 18040, 18037, 18034, 18032, 18029, 18026,
18024, 18021, 18019, 18016, 18013, 18011, 18008, 18005, 18003, 18000, 17998, 17995, 17992, 17990, 17987, 17984, 17982, 17979,
17977, 17974, 17971, 17969, 17966, 17963, 17961, 17958, 17956, 17953, 17950, 17948, 17945, 17942, 17940, 17937, 17934, 17932,
17929, 17927, 17924, 17921, 17919, 17916, 17913, 17911, 17908, 17906, 17903, 17900, 17898, 17895, 17892, 17890, 17887, 17885,
17882, 17879, 17877, 17874, 17871, 17869, 17866, 17863, 17861, 17858, 17856, 17853, 17850, 17848, 17845, 17842, 17840, 17837,
17834, 17832, 17829, 17827, 17824, 17821, 17819, 17816, 17813, 17811, 17808, 17805, 17803, 17800, 17798, 17795, 17792, 17790,
17787, 17784, 17782, 17779, 17776, 17774, 17771, 17769, 17766, 17763, 17761, 17758, 17755, 17753, 17750, 17747, 17745, 17742,
17739, 17737, 17734, 17732, 17729, 17726, 17724, 17721, 17718, 17716, 17713, 17710, 17708, 17705, 17702, 17700, 17697, 17695,
17692, 17689, 17687, 17684, 17681, 17679, 17676, 17673, 17671, 17668, 17665, 17663, 17660, 17658, 17655, 17652, 17650, 17647,
17644, 17642, 17639, 17636, 17634, 17631, 17628, 17626, 17623, 17620, 17618, 17615, 17613, 17610, 17607, 17605, 17602, 17599,
17597, 17594, 17591, 17589, 17586, 17583, 17581, 17578, 17575, 17573, 17570, 17567, 17565, 17562, 17559, 17557, 17554, 17552,
17549, 17546, 17544, 17541, 17538, 17536, 17533, 17530, 17528, 17525, 17522, 17520, 17517, 17514, 17512, 17509, 17506, 17504,
17501, 17498, 17496, 17493, 17490, 17488, 17485, 17483, 17480, 17477, 17475, 17472, 17469, 17467, 17464, 17461, 17459, 17456,
17453, 17451, 17448, 17445, 17443, 17440, 17437, 17435, 17432, 17429, 17427, 17424, 17421, 17419, 17416, 17413, 17411, 17408,
17405, 17403, 17400, 17397, 17395, 17392, 17389, 17387, 17384, 17381, 17379, 17376, 17373, 17371, 17368, 17365, 17363, 17360,
17357, 17355, 17352, 17349, 17347, 17344, 17341, 17339, 17336, 17333, 17331, 17328, 17325, 17323, 17320, 17317, 17315, 17312,
17309, 17307, 17304, 17301, 17299, 17296, 17293, 17291, 17288, 17285, 17283, 17280, 17277, 17275, 17272, 17269, 17267, 17264,
17261, 17259, 17256, 17253, 17251, 17248, 17245, 17243, 17240, 17237, 17235, 17232, 17229, 17227, 17224, 17221, 17219, 17216,
17213, 17211, 17208, 17205, 17203, 17200, 17197, 17195, 17192, 17189, 17187, 17184, 17181, 17179, 17176, 17173, 17171, 17168,
17165, 17163, 17160, 17157, 17154, 17152, 17149, 17146, 17144, 17141, 17138, 17136, 17133, 17130, 17128, 17125, 17122, 17120,
17117, 17114, 17112, 17109, 17106, 17104, 17101, 17098, 17096, 17093, 17090, 17088, 17085, 17082, 17079, 17077, 17074, 17071,
17069, 17066, 17063, 17061, 17058, 17055, 17053, 17050, 17047, 17045, 17042, 17039, 17037, 17034, 17031, 17028, 17026, 17023,
17020, 17018, 17015, 17012, 17010, 17007, 17004, 17002, 16999, 16996, 16994, 16991, 16988, 16986, 16983, 16980, 16977, 16975,
16972, 16969, 16967, 16964, 16961, 16959, 16956, 16953, 16951, 16948, 16945, 16943, 16940, 16937, 16934, 16932, 16929, 16926,
16924, 16921, 16918, 16916, 16913, 16910, 16908, 16905, 16902, 16899, 16897, 16894, 16891, 16889, 16886, 16883, 16881, 16878,
16875, 16873, 16870, 16867, 16864, 16862, 16859, 16856, 16854, 16851, 16848, 16846, 16843, 16840, 16838, 16835, 16832, 16829,
16827, 16824, 16821, 16819, 16816, 16813, 16811, 16808, 16805, 16802, 16800, 16797, 16794, 16792, 16789, 16786, 16784, 16781,
16778, 16776, 16773, 16770, 16767, 16765, 16762, 16759, 16757, 16754, 16751, 16749, 16746, 16743, 16740, 16738, 16735, 16732,
16730, 16727, 16724, 16722, 16719, 16716, 16713, 16711, 16708, 16705, 16703, 16700, 16697, 16694, 16692, 16689, 16686, 16684,
16681, 16678, 16676, 16673, 16670, 16667, 16665, 16662, 16659, 16657, 16654, 16651, 16648, 16646, 16643, 16640, 16638, 16635,
16632, 16630, 16627, 16624, 16621, 16619, 16616, 16613, 16611, 16608, 16605, 16602, 16600, 16597, 16594, 16592, 16589, 16586,
16584, 16581, 16578, 16575, 16573, 16570, 16567, 16565, 16562, 16559, 16556, 16554, 16551, 16548, 16546, 16543, 16540, 16537,
16535, 16532, 16529, 16527, 16524, 16521, 16518, 16516, 16513, 16510, 16508, 16505, 16502, 16499, 16497, 16494, 16491, 16489,
16486, 16483, 16480, 16478, 16475, 16472, 16470, 16467, 16464, 16461, 16459, 16456, 16453, 16451, 16448, 16445, 16442, 16440,
16437, 16434, 16432, 16429, 16426, 16423, 16421, 16418, 16415, 16413, 16410, 16407, 16404, 16402, 16399, 16396, 16393, 16391,
16388, 16385, 16383, 16380, 16377, 16374, 16372, 16369, 16366, 16364, 16361, 16358, 16355, 16353, 16350, 16347, 16344, 16342,

16339, 16336, 16334, 16331, 16328, 16325, 16323, 16320, 16317, 16315, 16312, 16309, 16306, 16304, 16301, 16298, 16295, 16293,
16290, 16287, 16285, 16282, 16279, 16276, 16274, 16271, 16268, 16265, 16263, 16260, 16257, 16255, 16252, 16249, 16246, 16244,
16241, 16238, 16235, 16233, 16230, 16227, 16225, 16222, 16219, 16216, 16214, 16211, 16208, 16205, 16203, 16200, 16197, 16195,
16192, 16189, 16186, 16184, 16181, 16178, 16175, 16173, 16170, 16167, 16164, 16162, 16159, 16156, 16154, 16151, 16148, 16145,
16143, 16140, 16137, 16134, 16132, 16129, 16126, 16123, 16121, 16118, 16115, 16113, 16110, 16107, 16104, 16102, 16099, 16096,
16093, 16091, 16088, 16085, 16082, 16080, 16077, 16074, 16071, 16069, 16066, 16063, 16061, 16058, 16055, 16052, 16050, 16047,
16044, 16041, 16039, 16036, 16033, 16030, 16028, 16025, 16022, 16019, 16017, 16014, 16011, 16008, 16006, 16003, 16000, 15997,
15995, 15992, 15989, 15987, 15984, 15981, 15978, 15976, 15973, 15970, 15967, 15965, 15962, 15959, 15956, 15954, 15951, 15948,
15945, 15943, 15940, 15937, 15934, 15932, 15929, 15926, 15923, 15921, 15918, 15915, 15912, 15910, 15907, 15904, 15901, 15899,
15896, 15893, 15890, 15888, 15885, 15882, 15879, 15877, 15874, 15871, 15868, 15866, 15863, 15860, 15857, 15855, 15852, 15849,
15846, 15844, 15841, 15838, 15835, 15833, 15830, 15827, 15824, 15822, 15819, 15816, 15813, 15811, 15808, 15805, 15802, 15800,
15797, 15794, 15791, 15789, 15786, 15783, 15780, 15778, 15775, 15772, 15769, 15767, 15764, 15761, 15758, 15756, 15753, 15750,
15747, 15745, 15742, 15739, 15736, 15734, 15731, 15728, 15725, 15723, 15720, 15717, 15714, 15712, 15709, 15706, 15703, 15701,
15698, 15695, 15692, 15689, 15687, 15684, 15681, 15678, 15676, 15673, 15670, 15667, 15665, 15662, 15659, 15656, 15654, 15651,
15648, 15645, 15643, 15640, 15637, 15634, 15632, 15629, 15626, 15623, 15620, 15618, 15615, 15612, 15609, 15607, 15604, 15601,
15598, 15596, 15593, 15590, 15587, 15585, 15582, 15579, 15576, 15574, 15571, 15568, 15565, 15562, 15560, 15557, 15554, 15551,
15549, 15546, 15543, 15540, 15538, 15535, 15532, 15529, 15527, 15524, 15521, 15518, 15515, 15513, 15510, 15507, 15504, 15502,
15499, 15496, 15493, 15491, 15488, 15485, 15482, 15479, 15477, 15474, 15471, 15468, 15466, 15463, 15460, 15457, 15455, 15452,
15449, 15446, 15443, 15441, 15438, 15435, 15432, 15430, 15427, 15424, 15421, 15419, 15416, 15413, 15410, 15407, 15405, 15402,
15399, 15396, 15394, 15391, 15388, 15385, 15382, 15380, 15377, 15374, 15371, 15369, 15366, 15363, 15360, 15357, 15355, 15352,
15349, 15346, 15344, 15341, 15338, 15335, 15333, 15330, 15327, 15324, 15321, 15319, 15316, 15313, 15310, 15308, 15305, 15302,
15299, 15296, 15294, 15291, 15288, 15285, 15283, 15280, 15277, 15274, 15271, 15269, 15266, 15263, 15260, 15257, 15255, 15252,
15249, 15246, 15244, 15241, 15238, 15235, 15232, 15230, 15227, 15224, 15221, 15219, 15216, 15213, 15210, 15207, 15205, 15202,
15199, 15196, 15194, 15191, 15188, 15185, 15182, 15180, 15177, 15174, 15171, 15168, 15166, 15163, 15160, 15157, 15155, 15152,
15149, 15146, 15143, 15141, 15138, 15135, 15132, 15129, 15127, 15124, 15121, 15118, 15116, 15113, 15110, 15107, 15104, 15102,
15099, 15096, 15093, 15090, 15088, 15085, 15082, 15079, 15076, 15074, 15071, 15068, 15065, 15063, 15060, 15057, 15054, 15051,
15049, 15046, 15043, 15040, 15037, 15035, 15032, 15029, 15026, 15023, 15021, 15018, 15015, 15012, 15010, 15007, 15004, 15001,
14998, 14996, 14993, 14990, 14987, 14984, 14982, 14979, 14976, 14973, 14970, 14968, 14965, 14962, 14959, 14956, 14954, 14951,
14948, 14945, 14942, 14940, 14937, 14934, 14931, 14928, 14926, 14923, 14920, 14917, 14914, 14912, 14909, 14906, 14903, 14900,
14898, 14895, 14892, 14889, 14886, 14884, 14881, 14878, 14875, 14873, 14870, 14867, 14864, 14861, 14859, 14856, 14853, 14850,
14847, 14845, 14842, 14839, 14836, 14833, 14830, 14828, 14825, 14822, 14819, 14816, 14814, 14811, 14808, 14805, 14802, 14800,
14797, 14794, 14791, 14788, 14786, 14783, 14780, 14777, 14774, 14772, 14769, 14766, 14763, 14760, 14758, 14755, 14752, 14749,
14746, 14744, 14741, 14738, 14735, 14732, 14730, 14727, 14724, 14721, 14718, 14716, 14713, 14710, 14707, 14704, 14701, 14699,
14696, 14693, 14690, 14687, 14685, 14682, 14679, 14676, 14673, 14671, 14668, 14665, 14662, 14659, 14657, 14654, 14651, 14648,
14645, 14642, 14640, 14637, 14634, 14631, 14628, 14626, 14623, 14620, 14617, 14614, 14612, 14609, 14606, 14603, 14600, 14598,
14595, 14592, 14589, 14586, 14583, 14581, 14578, 14575, 14572, 14569, 14567, 14564, 14561, 14558, 14555, 14552, 14550, 14547,
14544, 14541, 14538, 14536, 14533, 14530, 14527, 14524, 14522, 14519, 14516, 14513, 14510, 14507, 14505, 14502, 14499, 14496,
14493, 14491, 14488, 14485, 14482, 14479, 14476, 14474, 14471, 14468, 14465, 14462, 14460, 14457, 14454, 14451, 14448, 14445,
14443, 14440, 14437, 14434, 14431, 14429, 14426, 14423, 14420, 14417, 14414, 14412, 14409, 14406, 14403, 14400, 14397, 14395,
14392, 14389, 14386, 14383, 14381, 14378, 14375, 14372, 14369, 14366, 14364, 14361, 14358, 14355, 14352, 14349, 14347, 14344,
14341, 14338, 14335, 14333, 14330, 14327, 14324, 14321, 14318, 14316, 14313, 14310, 14307, 14304, 14301, 14299, 14296, 14293,
14290, 14287, 14284, 14282, 14279, 14276, 14273, 14270, 14268, 14265, 14262, 14259, 14256, 14253, 14251, 14248, 14245, 14242,
14239, 14236, 14234, 14231, 14228, 14225, 14222, 14219, 14217, 14214, 14211, 14208, 14205, 14202, 14200, 14197, 14194, 14191,
14188, 14185, 14183, 14180, 14177, 14174, 14171, 14168, 14166, 14163, 14160, 14157, 14154, 14151, 14149, 14146, 14143, 14140,
14137, 14134, 14132, 14129, 14126, 14123, 14120, 14117, 14115, 14112, 14109, 14106, 14103, 14100, 14098, 14095, 14092, 14089,
14086, 14083, 14081, 14078, 14075, 14072, 14069, 14066, 14064, 14061, 14058, 14055, 14052, 14049, 14047, 14044, 14041, 14038,
14035, 14032, 14030, 14027, 14024, 14021, 14018, 14015, 14012, 14010, 14007, 14004, 14001, 13998, 13995, 13993, 13990, 13987,
13984, 13981, 13978, 13976, 13973, 13970, 13967, 13964, 13961, 13958, 13956, 13953, 13950, 13947, 13944, 13941, 13939, 13936,
13933, 13930, 13927, 13924, 13922, 13919, 13916, 13913, 13910, 13907, 13904, 13902, 13899, 13896, 13893, 13890, 13887, 13885,
13882, 13879, 13876, 13873, 13870, 13867, 13865, 13862, 13859, 13856, 13853, 13850, 13848, 13845, 13842, 13839, 13836, 13833,
13830, 13828, 13825, 13822, 13819, 13816, 13813, 13811, 13808, 13805, 13802, 13799, 13796, 13793, 13791, 13788, 13785, 13782,
13779, 13776, 13773, 13771, 13768, 13765, 13762, 13759, 13756, 13754, 13751, 13748, 13745, 13742, 13739, 13736, 13734, 13731,
13728, 13725, 13722, 13719, 13716, 13714, 13711, 13708, 13705, 13702, 13699, 13696, 13694, 13691, 13688, 13685, 13682, 13679,
13676, 13674, 13671, 13668, 13665, 13662, 13659, 13656, 13654, 13651, 13648, 13645, 13642, 13639, 13636, 13634, 13631, 13628,
13625, 13622, 13619, 13616, 13614, 13611, 13608, 13605, 13602, 13599, 13596, 13594, 13591, 13588, 13585, 13582, 13579, 13576,
13574, 13571, 13568, 13565, 13562, 13559, 13556, 13554, 13551, 13548, 13545, 13542, 13539, 13536, 13534, 13531, 13528, 13525,
13522, 13519, 13516, 13514, 13511, 13508, 13505, 13502, 13499, 13496, 13494, 13491, 13488, 13485, 13482, 13479, 13476, 13473,
13471, 13468, 13465, 13462, 13459, 13456, 13453, 13451, 13448, 13445, 13442, 13439, 13436, 13433, 13430, 13428, 13425, 13422,
13419, 13416, 13413, 13410, 13408, 13405, 13402, 13399, 13396, 13393, 13390, 13387, 13385, 13382, 13379, 13376, 13373, 13370,
13367, 13365, 13362, 13359, 13356, 13353, 13350, 13347, 13344, 13342, 13339, 13336, 13333, 13330, 13327, 13324, 13322, 13319,
13316, 13313, 13310, 13307, 13304, 13301, 13299, 13296, 13293, 13290, 13287, 13284, 13281, 13278, 13276, 13273, 13270, 13267,
13264, 13261, 13258, 13255, 13253, 13250, 13247, 13244, 13241, 13238, 13235, 13232, 13230, 13227, 13224, 13221, 13218, 13215,
13212, 13209, 13207, 13204, 13201, 13198, 13195, 13192, 13189, 13186, 13184, 13181, 13178, 13175, 13172, 13169, 13166, 13163,
13161, 13158, 13155, 13152, 13149, 13146, 13143, 13140, 13138, 13135, 13132, 13129, 13126, 13123, 13120, 13117, 13115, 13112,
13109, 13106, 13103, 13100, 13097, 13094, 13092, 13089, 13086, 13083, 13080, 13077, 13074, 13071, 13068, 13066, 13063, 13060,
13057, 13054, 13051, 13048, 13045, 13043, 13040, 13037, 13034, 13031, 13028, 13025, 13022, 13019, 13017, 13014, 13011, 13008,
13005, 13002, 12999, 12996, 12994, 12991, 12988, 12985, 12982, 12979, 12976, 12973, 12970, 12968, 12965, 12962, 12959, 12956,
12953, 12950, 12947, 12944, 12942, 12939, 12936, 12933, 12930, 12927, 12924, 12921, 12918, 12916, 12913, 12910, 12907, 12904,
12901, 12898, 12895, 12893, 12890, 12887, 12884, 12881, 12878, 12875, 12872, 12869, 12867, 12864, 12861, 12858, 12855, 12852,
12849, 12846, 12843, 12840, 12838, 12835, 12832, 12829, 12826, 12823, 12820, 12817, 12814, 12812, 12809, 12806, 12803, 12800,
12797, 12794, 12791, 12788, 12786, 12783, 12780, 12777, 12774, 12771, 12768, 12765, 12762, 12760, 12757, 12754, 12751, 12748,

12745, 12742, 12739, 12736, 12733, 12731, 12728, 12725, 12722, 12719, 12716, 12713, 12710, 12707, 12705, 12702, 12699, 12696, 12693, 12690, 12687, 12684, 12681, 12678, 12676, 12673, 12670, 12667, 12664, 12661, 12658, 12655, 12652, 12649, 12647, 12644, 12641, 12638, 12635, 12632, 12629, 12626, 12623, 12620, 12618, 12615, 12612, 12609, 12606, 12603, 12600, 12597, 12594, 12591, 12589, 12586, 12583, 12580, 12577, 12574, 12571, 12568, 12565, 12562, 12560, 12557, 12554, 12551, 12548, 12545, 12542, 12539, 12536, 12533, 12531, 12528, 12525, 12522, 12519, 12516, 12513, 12510, 12507, 12504, 12502, 12499, 12496, 12493, 12490, 12487, 12484, 12481, 12478, 12475, 12472, 12470, 12467, 12464, 12461, 12458, 12455, 12452, 12449, 12446, 12443, 12441, 12438, 12435, 12432, 12429, 12426, 12423, 12420, 12417, 12414, 12411, 12409, 12406, 12403, 12400, 12397, 12394, 12391, 12388, 12385, 12382, 12379, 12377, 12374, 12371, 12368, 12365, 12362, 12359, 12356, 12353, 12350, 12347, 12345, 12342, 12339, 12336, 12333, 12330, 12327, 12324, 12321, 12318, 12315, 12313, 12310, 12307, 12304, 12301, 12298, 12295, 12292, 12289, 12286, 12283, 12280, 12278, 12275, 12272, 12269, 12266, 12263, 12260, 12257, 12254, 12251, 12248, 12246, 12243, 12240, 12237, 12234, 12231, 12228, 12225, 12222, 12219, 12216, 12213, 12211, 12208, 12205, 12202, 12199, 12196, 12193, 12190, 12187, 12184, 12181, 12178, 12176, 12173, 12170, 12167, 12164, 12161, 12158, 12155, 12152, 12149, 12146, 12143, 12141, 12138, 12135, 12132, 12129, 12126, 12123, 12120, 12117, 12114, 12111, 12108, 12106, 12103, 12100, 12097, 12094, 12091, 12088, 12085, 12082, 12079, 12076, 12073, 12071, 12068, 12065, 12062, 12059, 12056, 12053, 12050, 12047, 12044, 12041, 12038, 12035, 12033, 12030, 12027, 12024, 12021, 12018, 12015, 12012, 12009, 12006, 12003, 12000, 11997, 11995, 11992, 11989, 11986, 11983, 11980, 11977, 11974, 11971, 11968, 11965, 11962, 11959, 11957, 11954, 11951, 11948, 11945, 11942, 11939, 11936, 11933, 11930, 11927, 11924, 11921, 11918, 11916, 11913, 11910, 11907, 11904, 11901, 11898, 11895, 11892, 11889, 11886, 11883, 11880, 11877, 11875, 11872, 11869, 11866, 11863, 11860, 11857, 11854, 11851, 11848, 11845, 11842, 11839, 11836, 11834, 11831, 11828, 11825, 11822, 11819, 11816, 11813, 11810, 11807, 11804, 11801, 11798, 11795, 11793, 11790, 11787, 11784, 11781, 11778, 11775, 11772, 11769, 11766, 11763, 11760, 11757, 11754, 11751, 11749, 11746, 11743, 11740, 11737, 11734, 11731, 11728, 11725, 11722, 11719, 11716, 11713, 11710, 11707, 11705, 11702, 11699, 11696, 11693, 11690, 11687, 11684, 11681, 11678, 11675, 11672, 11669, 11666, 11663, 11661, 11658, 11655, 11652, 11649, 11646, 11643, 11640, 11637, 11634, 11631, 11628, 11625, 11622, 11619, 11616, 11614, 11611, 11608, 11605, 11602, 11599, 11596, 11593, 11590, 11587, 11584, 11581, 11578, 11575, 11572, 11569, 11567, 11564, 11561, 11558, 11555, 11552, 11549, 11546, 11543, 11540, 11537, 11534, 11531, 11528, 11525, 11522, 11519, 11517, 11514, 11511, 11508, 11505, 11502, 11499, 11496, 11493, 11490, 11487, 11484, 11481, 11478, 11475, 11472, 11469, 11467, 11464, 11461, 11458, 11455, 11452, 11449, 11446, 11443, 11440, 11437, 11434, 11431, 11428, 11425, 11422, 11419, 11416, 11414, 11411, 11408, 11405, 11402, 11399, 11396, 11393, 11390, 11387, 11384, 11381, 11378, 11375, 11372, 11369, 11366, 11363, 11361, 11358, 11355, 11352, 11349, 11346, 11343, 11340, 11337, 11334, 11331, 11328, 11325, 11322, 11319, 11316, 11313, 11310, 11307, 11305, 11302, 11299, 11296, 11293, 11290, 11287, 11284, 11281, 11278, 11275, 11272, 11269, 11266, 11263, 11260, 11257, 11254, 11251, 11248, 11246, 11243, 11240, 11237, 11234, 11231, 11228, 11225, 11222, 11219, 11216, 11213, 11210, 11207, 11204, 11201, 11198, 11195, 11192, 11189, 11186, 11184, 11181, 11178, 11175, 11172, 11169, 11166, 11163, 11160, 11157, 11154, 11151, 11148, 11145, 11142, 11139, 11136, 11133, 11130, 11127, 11124, 11121, 11119, 11116, 11113, 11110, 11107, 11104, 11101, 11098, 11095, 11092, 11089, 11086, 11083, 11080, 11077, 11074, 11071, 11068, 11065, 11062, 11059, 11056, 11053, 11051, 11048, 11045, 11042, 11039, 11036, 11033, 11030, 11027, 11024, 11021, 11018, 11015, 11012, 11009, 11006, 11003, 11000, 10997, 10994, 10991, 10988, 10985, 10982, 10980, 10977, 10974, 10971, 10968, 10965, 10962, 10959, 10956, 10953, 10950, 10947, 10944, 10941, 10938, 10935, 10932, 10929, 10926, 10923, 10920, 10917, 10914, 10911, 10908, 10905, 10903, 10900, 10897, 10894, 10891, 10888, 10885, 10882, 10879, 10876, 10873, 10870, 10867, 10864, 10861, 10858, 10855, 10852, 10849, 10846, 10843, 10840, 10837, 10834, 10831, 10828, 10825, 10823, 10820, 10817, 10814, 10811, 10808, 10805, 10802, 10799, 10796, 10793, 10790, 10787, 10784, 10781, 10778, 10775, 10772, 10769, 10766, 10763, 10760, 10757, 10754, 10751, 10748, 10745, 10742, 10739, 10736, 10734, 10731, 10728, 10725, 10722, 10719, 10716, 10713, 10710, 10707, 10704, 10701, 10698, 10695, 10692, 10689, 10686, 10683, 10680, 10677, 10674, 10671, 10668, 10665, 10662, 10659, 10656, 10653, 10650, 10647, 10644, 10641, 10638, 10635, 10633, 10630, 10627, 10624, 10621, 10618, 10615, 10612, 10609, 10606, 10603, 10600, 10597, 10594, 10591, 10588, 10585, 10582, 10579, 10576, 10573, 10570, 10567, 10564, 10561, 10558, 10555, 10552, 10549, 10546, 10543, 10540, 10537, 10534, 10531, 10528, 10525, 10523, 10520, 10517, 10514, 10511, 10508, 10505, 10502, 10499, 10496, 10493, 10490, 10487, 10484, 10481, 10478, 10475, 10472, 10469, 10466, 10463, 10460, 10457, 10454, 10451, 10448, 10445, 10442, 10439, 10436, 10433, 10430, 10427, 10424, 10421, 10418, 10415, 10412, 10409, 10406, 10403, 10400, 10397, 10394, 10392, 10389, 10386, 10383, 10380, 10377, 10374, 10371, 10368, 10365, 10362, 10359, 10356, 10353, 10350, 10347, 10344, 10341, 10338, 10335, 10332, 10329, 10326, 10323, 10320, 10317, 10314, 10311, 10308, 10305, 10302, 10299, 10296, 10293, 10290, 10287, 10284, 10281, 10278, 10275, 10272, 10269, 10266, 10263, 10260, 10257, 10254, 10251, 10248, 10245, 10242, 10239, 10236, 10233, 10230, 10227, 10225, 10222, 10219, 10216, 10213, 10210, 10207, 10204, 10201, 10198, 10195, 10192, 10189, 10186, 10183, 10180, 10177, 10174, 10171, 10168, 10165, 10162, 10159, 10156, 10153, 10150, 10147, 10144, 10141, 10138, 10135, 10132, 10129, 10126, 10123, 10120, 10117, 10114, 10111, 10108, 10105, 10102, 10099, 10096, 10093, 10090, 10087, 10084, 10081, 10078, 10075, 10072, 10069, 10066, 10063, 10060, 10057, 10054, 10051, 10048, 10045, 10042, 10039, 10036, 10033, 10030, 10027, 10024, 10021, 10018, 10015, 10012, 10009, 10006, 10003, 10000, 9997, 9994, 9991, 9988, 9985, 9982, 9979, 9976, 9973, 9970, 9967, 9964, 9961, 9959, 9956, 9953, 9950, 9947, 9944, 9941, 9938, 9935, 9932, 9929, 9926, 9923, 9920, 9917, 9914, 9911, 9908, 9905, 9902, 9899, 9896, 9893, 9890, 9887, 9884, 9881, 9878, 9875, 9872, 9869, 9866, 9863, 9860, 9857, 9854, 9851, 9848, 9845, 9842, 9839, 9836, 9833, 9830, 9827, 9824, 9821, 9818, 9815, 9812, 9809, 9806, 9803, 9800, 9797, 9794, 9791, 9788, 9785, 9782, 9779, 9776, 9773, 9770, 9767, 9764, 9761, 9758, 9755, 9752, 9749, 9746, 9743, 9740, 9737, 9734, 9731, 9728, 9725, 9722, 9719, 9716, 9713, 9710, 9707, 9704, 9701, 9698, 9695, 9692, 9689, 9686, 9683, 9680, 9677, 9674, 9671, 9668, 9665, 9662, 9659, 9656, 9653, 9650, 9647, 9644, 9641, 9638, 9635, 9632, 9629, 9626, 9623, 9620, 9617, 9614, 9611, 9608, 9605, 9602, 9599, 9596, 9593, 9590, 9587, 9584, 9581, 9578, 9575, 9572, 9569, 9566, 9563, 9560, 9557, 9554, 9551, 9548, 9545, 9542, 9539, 9536, 9533, 9530, 9527, 9524, 9521, 9518, 9515, 9512, 9509, 9506, 9503, 9500, 9497, 9494, 9491, 9487, 9484, 9481, 9478, 9475, 9472, 9469, 9466, 9463, 9460, 9457, 9454, 9451, 9448, 9445, 9442, 9439, 9436, 9433, 9430, 9427, 9424, 9421, 9418, 9415, 9412, 9409, 9406, 9403, 9400, 9397, 9394, 9391, 9388, 9385, 9382, 9379, 9376, 9373, 9370, 9367, 9364, 9361, 9358, 9355, 9352, 9349, 9346, 9343, 9340, 9337, 9334, 9331, 9328, 9325, 9322, 9319, 9316, 9313, 9310, 9307, 9304, 9301, 9298, 9295, 9292, 9289, 9286, 9283, 9280, 9277, 9274, 9271, 9268, 9265, 9262, 9259, 9256, 9253, 9250, 9247, 9244, 9241, 9238, 9235, 9232, 9229, 9226, 9223, 9220, 9217, 9213, 9210, 9207, 9204, 9201, 9198, 9195, 9192, 9189, 9186, 9183, 9180, 9177, 9174, 9171, 9168, 9165, 9162, 9159, 9156, 9153, 9150, 9147, 9144, 9141, 9138, 9135, 9132, 9129, 9126, 9123, 9120, 9117, 9114, 9111, 9108, 9105, 9102, 9099, 9096, 9093, 9090, 9087, 9084, 9081, 9078, 9075, 9072, 9069, 9066, 9063, 9060, 9057, 9054, 9051, 9048, 9045, 9042, 9038, 9035, 9032, 9029, 9026, 9023, 9020, 9017, 9014, 9011, 9008, 9005, 9002, 8999, 8996, 8993, 8990, 8987, 8984, 8981, 8978, 8975, 8972, 8969, 8966, 8963, 8960, 8957, 8954, 8951, 8948, 8945, 8942, 8939, 8936, 8933, 8930, 8927, 8924, 8921, 8918, 8915, 8912, 8909, 8906, 8903, 8900, 8896, 8893, 8890, 8887, 8884, 8881, 8878, 8875, 8872, 8869, 8866, 8863, 8860, 8857, 8854, 8851, 8848, 8845, 8842, 8839, 8836, 8833, 8830, 8827, 8824, 8821, 8818, 8815, 8812, 8809, 8806, 8803, 8800, 8797, 8794, 8791,

8788, 8785, 8782, 8778, 8775, 8772, 8769, 8766, 8763, 8760, 8757, 8754, 8751, 8748, 8745, 8742, 8739, 8736, 8733, 8730, 8727, 8724, 8721, 8718, 8715, 8712, 8709, 8706, 8703, 8700, 8697, 8694, 8691, 8688, 8685, 8682, 8679, 8676, 8673, 8669, 8666, 8663, 8660, 8657, 8654, 8651, 8648, 8645, 8642, 8639, 8636, 8633, 8630, 8627, 8624, 8621, 8618, 8615, 8612, 8609, 8606, 8603, 8600, 8597, 8594, 8591, 8588, 8585, 8582, 8579, 8576, 8572, 8569, 8566, 8563, 8560, 8557, 8554, 8551, 8548, 8545, 8542, 8539, 8536, 8533, 8530, 8527, 8524, 8521, 8518, 8515, 8512, 8509, 8506, 8503, 8500, 8497, 8494, 8491, 8488, 8485, 8481, 8478, 8475, 8472, 8469, 8466, 8463, 8460, 8457, 8454, 8451, 8448, 8445, 8442, 8439, 8436, 8433, 8430, 8427, 8424, 8421, 8418, 8415, 8412, 8409, 8406, 8403, 8400, 8396, 8393, 8390, 8387, 8384, 8381, 8378, 8375, 8372, 8369, 8366, 8363, 8360, 8357, 8354, 8351, 8348, 8345, 8342, 8339, 8336, 8333, 8330, 8327, 8324, 8321, 8318, 8314, 8311, 8308, 8305, 8302, 8299, 8296, 8293, 8290, 8287, 8284, 8281, 8278, 8275, 8272, 8269, 8266, 8263, 8260, 8257, 8254, 8251, 8248, 8245, 8242, 8238, 8235, 8232, 8229, 8226, 8223, 8220, 8217, 8214, 8211, 8208, 8205, 8202, 8199, 8196, 8193, 8190, 8187, 8184, 8181, 8178, 8175, 8172, 8169, 8165, 8162, 8159, 8156, 8153, 8150, 8147, 8144, 8141, 8138, 8135, 8132, 8129, 8126, 8123, 8120, 8117, 8114, 8111, 8108, 8105, 8102, 8099, 8095, 8092, 8089, 8086, 8083, 8080, 8077, 8074, 8071, 8068, 8065, 8062, 8059, 8056, 8053, 8050, 8047, 8044, 8041, 8038, 8035, 8032, 8029, 8025, 8022, 8019, 8016, 8013, 8010, 8007, 8004, 8001, 7998, 7995, 7992, 7989, 7986, 7983, 7980, 7977, 7974, 7971, 7968, 7965, 7961, 7958, 7955, 7952, 7949, 7946, 7943, 7940, 7937, 7934, 7931, 7928, 7925, 7922, 7919, 7916, 7913, 7910, 7907, 7904, 7901, 7897, 7894, 7891, 7888, 7885, 7882, 7879, 7876, 7873, 7870, 7867, 7864, 7861, 7858, 7855, 7852, 7849, 7846, 7843, 7840, 7836, 7833, 7830, 7827, 7824, 7821, 7818, 7815, 7812, 7809, 7806, 7803, 7800, 7797, 7794, 7791, 7788, 7785, 7782, 7778, 7775, 7772, 7769, 7766, 7763, 7760, 7757, 7754, 7751, 7748, 7745, 7742, 7739, 7736, 7733, 7730, 7727, 7724, 7721, 7717, 7714, 7711, 7708, 7705, 7702, 7699, 7696, 7693, 7690, 7687, 7684, 7681, 7678, 7675, 7672, 7669, 7666, 7662, 7659, 7656, 7653, 7650, 7647, 7644, 7641, 7638, 7635, 7632, 7629, 7626, 7623, 7620, 7617, 7614, 7611, 7607, 7604, 7601, 7598, 7595, 7592, 7589, 7586, 7583, 7580, 7577, 7574, 7571, 7568, 7565, 7562, 7559, 7556, 7552, 7549, 7546, 7543, 7540, 7537, 7534, 7531, 7528, 7525, 7522, 7519, 7516, 7513, 7510, 7507, 7504, 7500, 7497, 7494, 7491, 7488, 7485, 7482, 7479, 7476, 7473, 7470, 7467, 7464, 7461, 7458, 7455, 7452, 7448, 7445, 7442, 7439, 7436, 7433, 7430, 7427, 7424, 7421, 7418, 7415, 7412, 7409, 7406, 7403, 7400, 7396, 7393, 7390, 7387, 7384, 7381, 7378, 7375, 7372, 7369, 7366, 7363, 7360, 7357, 7354, 7351, 7347, 7344, 7341, 7338, 7335, 7332, 7329, 7326, 7323, 7320, 7317, 7314, 7311, 7308, 7305, 7302, 7299, 7295, 7292, 7289, 7286, 7283, 7280, 7277, 7274, 7271, 7268, 7265, 7262, 7259, 7256, 7253, 7249, 7246, 7243, 7240, 7237, 7234, 7231, 7228, 7225, 7222, 7219, 7216, 7213, 7210, 7207, 7204, 7200, 7197, 7194, 7191, 7188, 7185, 7182, 7179, 7176, 7173, 7170, 7167, 7164, 7161, 7158, 7154, 7151, 7148, 7145, 7142, 7139, 7136, 7133, 7130, 7127, 7124, 7121, 7118, 7115, 7112, 7108, 7105, 7102, 7099, 7096, 7093, 7090, 7087, 7084, 7081, 7078, 7075, 7072, 7069, 7066, 7062, 7059, 7056, 7053, 7050, 7047, 7044, 7041, 7038, 7035, 7032, 7029, 7026, 7023, 7020, 7016, 7013, 7010, 7007, 7004, 7001, 6998, 6995, 6992, 6989, 6986, 6983, 6980, 6977, 6973, 6970, 6967, 6964, 6961, 6958, 6955, 6952, 6949, 6946, 6943, 6940, 6937, 6934, 6931, 6927, 6924, 6921, 6918, 6915, 6912, 6909, 6906, 6903, 6900, 6897, 6894, 6891, 6888, 6884, 6881, 6878, 6875, 6872, 6869, 6866, 6863, 6860, 6857, 6854, 6851, 6848, 6845, 6841, 6838, 6835, 6832, 6829, 6826, 6823, 6820, 6817, 6814, 6811, 6808, 6805, 6801, 6798, 6795, 6792, 6789, 6786, 6783, 6780, 6777, 6774, 6771, 6768, 6765, 6762, 6758, 6755, 6752, 6749, 6746, 6743, 6740, 6737, 6734, 6731, 6728, 6725, 6722, 6718, 6715, 6712, 6709, 6706, 6703, 6700, 6697, 6694, 6691, 6688, 6685, 6682, 6679, 6675, 6672, 6669, 6666, 6663, 6660, 6657, 6654, 6651, 6648, 6645, 6642, 6639, 6635, 6632, 6629, 6626, 6623, 6620, 6617, 6614, 6611, 6608, 6605, 6602, 6599, 6595, 6592, 6589, 6586, 6583, 6580, 6577, 6574, 6571, 6568, 6565, 6562, 6559, 6555, 6552, 6549, 6546, 6543, 6540, 6537, 6534, 6531, 6528, 6525, 6522, 6519, 6515, 6512, 6509, 6506, 6503, 6500, 6497, 6494, 6491, 6488, 6485, 6482, 6478, 6475, 6472, 6469, 6466, 6463, 6460, 6457, 6454, 6451, 6448, 6445, 6442, 6438, 6435, 6432, 6429, 6426, 6423, 6420, 6417, 6414, 6411, 6408, 6405, 6401, 6398, 6395, 6392, 6389, 6386, 6383, 6380, 6377, 6374, 6371, 6368, 6364, 6361, 6358, 6355, 6352, 6349, 6346, 6343, 6340, 6337, 6334, 6331, 6328, 6324, 6321, 6318, 6315, 6312, 6309, 6306, 6303, 6300, 6297, 6294, 6291, 6287, 6284, 6281, 6278, 6275, 6272, 6269, 6266, 6263, 6260, 6257, 6254, 6250, 6247, 6244, 6241, 6238, 6235, 6232, 6229, 6226, 6223, 6220, 6216, 6213, 6210, 6207, 6204, 6201, 6198, 6195, 6192, 6189, 6186, 6183, 6179, 6176, 6173, 6170, 6167, 6164, 6161, 6158, 6155, 6152, 6149, 6146, 6142, 6139, 6136, 6133, 6130, 6127, 6124, 6121, 6118, 6115, 6112, 6109, 6105, 6102, 6099, 6096, 6093, 6090, 6087, 6084, 6081, 6078, 6075, 6071, 6068, 6065, 6062, 6059, 6056, 6053, 6050, 6047, 6044, 6041, 6037, 6034, 6031, 6028, 6025, 6022, 6019, 6016, 6013, 6010, 6007, 6004, 6000, 5997, 5994, 5991, 5988, 5985, 5982, 5979, 5976, 5973, 5970, 5966, 5963, 5960, 5957, 5954, 5951, 5948, 5945, 5942, 5939, 5936, 5932, 5929, 5926, 5923, 5920, 5917, 5914, 5911, 5908, 5905, 5902, 5898, 5895, 5892, 5889, 5886, 5883, 5880, 5877, 5874, 5871, 5868, 5864, 5861, 5858, 5855, 5852, 5849, 5846, 5843, 5840, 5837, 5834, 5830, 5827, 5824, 5821, 5818, 5815, 5812, 5809, 5806, 5803, 5800, 5796, 5793, 5790, 5787, 5784, 5781, 5778, 5775, 5772, 5769, 5766, 5762, 5759, 5756, 5753, 5750, 5747, 5744, 5741, 5738, 5735, 5732, 5728, 5725, 5722, 5719, 5716, 5713, 5710, 5707, 5704, 5701, 5698, 5694, 5691, 5688, 5685, 5682, 5679, 5676, 5673, 5670, 5667, 5663, 5660, 5657, 5654, 5651, 5648, 5645, 5642, 5639, 5636, 5633, 5629, 5626, 5623, 5620, 5617, 5614, 5611, 5608, 5605, 5602, 5598, 5595, 5592, 5589, 5586, 5583, 5580, 5577, 5574, 5571, 5568, 5564, 5561, 5558, 5555, 5552, 5549, 5546, 5543, 5540, 5537, 5533, 5530, 5527, 5524, 5521, 5518, 5515, 5512, 5509, 5506, 5503, 5499, 5496, 5493, 5490, 5487, 5484, 5481, 5478, 5475, 5472, 5468, 5465, 5462, 5459, 5456, 5453, 5450, 5447, 5444, 5441, 5437, 5434, 5431, 5428, 5425, 5422, 5419, 5416, 5413, 5410, 5406, 5403, 5400, 5397, 5394, 5391, 5388, 5385, 5382, 5379, 5375, 5372, 5369, 5366, 5363, 5360, 5357, 5354, 5351, 5348, 5344, 5341, 5338, 5335, 5332, 5329, 5326, 5323, 5320, 5317, 5313, 5310, 5307, 5304, 5301, 5298, 5295, 5292, 5289, 5286, 5282, 5279, 5276, 5273, 5270, 5267, 5264, 5261, 5258, 5255, 5251, 5248, 5245, 5242, 5239, 5236, 5233, 5230, 5227, 5224, 5220, 5217, 5214, 5211, 5208, 5205, 5202, 5199, 5196, 5193, 5189, 5186, 5183, 5180, 5177, 5174, 5171, 5168, 5165, 5162, 5158, 5155, 5152, 5149, 5146, 5143, 5140, 5137, 5134, 5131, 5127, 5124, 5121, 5118, 5115, 5112, 5109, 5106, 5103, 5099, 5096, 5093, 5090, 5087, 5084, 5081, 5078, 5075, 5072, 5068, 5065, 5062, 5059, 5056, 5053, 5050, 5047, 5044, 5041, 5037, 5034, 5031, 5028, 5025, 5022, 5019, 5016, 5013, 5009, 5006, 5003, 5000, 4997, 4994, 4991, 4988, 4985, 4982, 4978, 4975, 4972, 4969, 4966, 4963, 4960, 4957, 4954, 4950, 4947, 4944, 4941, 4938, 4935, 4932, 4929, 4926, 4923, 4919, 4916, 4913, 4910, 4907, 4904, 4901, 4898, 4895, 4891, 4888, 4885, 4882, 4879, 4876, 4873, 4870, 4867, 4863, 4860, 4857, 4854, 4851, 4848, 4845, 4842, 4839, 4836, 4832, 4829, 4826, 4823, 4820, 4817, 4814, 4811, 4808, 4804, 4801, 4798, 4795, 4792, 4789, 4786, 4783, 4780, 4776, 4773, 4770, 4767, 4764, 4761, 4758, 4755, 4752, 4749, 4745, 4742, 4739, 4736, 4733, 4730, 4727, 4724, 4721, 4717, 4714, 4711, 4708, 4705, 4702, 4699, 4696, 4693, 4689, 4686, 4683, 4680, 4677, 4674, 4671, 4668, 4665, 4661, 4658, 4655, 4652, 4649, 4646, 4643, 4640, 4637, 4633, 4630, 4627, 4624, 4621, 4618, 4615, 4612, 4609, 4605, 4602, 4599, 4596, 4593, 4590, 4587, 4584, 4581, 4577, 4574, 4571, 4568, 4565, 4562, 4559, 4556, 4553, 4549, 4546, 4543, 4540, 4537, 4534, 4531, 4528, 4525, 4521, 4518, 4515, 4512, 4509, 4506, 4503, 4500, 4497, 4493, 4490, 4487, 4484, 4481, 4478, 4475, 4472, 4469, 4465, 4462, 4459, 4456, 4453, 4450, 4447, 4444, 4441, 4437, 4434, 4431, 4428, 4425, 4422, 4419, 4416, 4413, 4409, 4406, 4403, 4400, 4397, 4394, 4391, 4388, 4385, 4381, 4378, 4375, 4372, 4369, 4366, 4363, 4360, 4357, 4353, 4350, 4347, 4344, 4341, 4338, 4335, 4332, 4328, 4325, 4322, 4319, 4316, 4313, 4310, 4307, 4304, 4300, 4297, 4294, 4291, 4288, 4285, 4282, 4279, 4276, 4272, 4269, 4266, 4263, 4260, 4257, 4254, 4251, 4248, 4244, 4241, 4238, 4235, 4232, 4229, 4226, 4223, 4219, 4216, 4213, 4210, 4207, 4204,

4201, 4198, 4195, 4191, 4188, 4185, 4182, 4179, 4176, 4173, 4170, 4167, 4163, 4160, 4157, 4154, 4151, 4148, 4145, 4142, 4138,
4135, 4132, 4129, 4126, 4123, 4120, 4117, 4114, 4110, 4107, 4104, 4101, 4098, 4095, 4092, 4089, 4085, 4082, 4079, 4076, 4073,
4070, 4067, 4064, 4061, 4057, 4054, 4051, 4048, 4045, 4042, 4039, 4036, 4032, 4029, 4026, 4023, 4020, 4017, 4014, 4011, 4008,
4004, 4001, 3998, 3995, 3992, 3989, 3986, 3983, 3979, 3976, 3973, 3970, 3967, 3964, 3961, 3958, 3955, 3951, 3948, 3945, 3942,
3939, 3936, 3933, 3930, 3926, 3923, 3920, 3917, 3914, 3911, 3908, 3905, 3902, 3898, 3895, 3892, 3889, 3886, 3883, 3880, 3877,
3873, 3870, 3867, 3864, 3861, 3858, 3855, 3852, 3848, 3845, 3842, 3839, 3836, 3833, 3830, 3827, 3824, 3820, 3817, 3814, 3811,
3808, 3805, 3802, 3799, 3795, 3792, 3789, 3786, 3783, 3780, 3777, 3774, 3770, 3767, 3764, 3761, 3758, 3755, 3752, 3749, 3745,
3742, 3739, 3736, 3733, 3730, 3727, 3724, 3721, 3717, 3714, 3711, 3708, 3705, 3702, 3699, 3696, 3692, 3689, 3686, 3683, 3680,
3677, 3674, 3671, 3667, 3664, 3661, 3658, 3655, 3652, 3649, 3646, 3642, 3639, 3636, 3633, 3630, 3627, 3624, 3621, 3618, 3614,
3611, 3608, 3605, 3602, 3599, 3596, 3593, 3589, 3586, 3583, 3580, 3577, 3574, 3571, 3568, 3564, 3561, 3558, 3555, 3552, 3549,
3546, 3543, 3539, 3536, 3533, 3530, 3527, 3524, 3521, 3518, 3514, 3511, 3508, 3505, 3502, 3499, 3496, 3493, 3489, 3486, 3483,
3480, 3477, 3474, 3471, 3468, 3464, 3461, 3458, 3455, 3452, 3449, 3446, 3443, 3439, 3436, 3433, 3430, 3427, 3424, 3421, 3418,
3414, 3411, 3408, 3405, 3402, 3399, 3396, 3393, 3389, 3386, 3383, 3380, 3377, 3374, 3371, 3368, 3364, 3361, 3358, 3355, 3352,
3349, 3346, 3343, 3339, 3336, 3333, 3330, 3327, 3324, 3321, 3318, 3314, 3311, 3308, 3305, 3302, 3299, 3296, 3293, 3289, 3286,
3283, 3280, 3277, 3274, 3271, 3268, 3264, 3261, 3258, 3255, 3252, 3249, 3246, 3243, 3239, 3236, 3233, 3230, 3227, 3224, 3221,
3218, 3214, 3211, 3208, 3205, 3202, 3199, 3196, 3193, 3189, 3186, 3183, 3180, 3177, 3174, 3171, 3168, 3164, 3161, 3158, 3155,
3152, 3149, 3146, 3143, 3139, 3136, 3133, 3130, 3127, 3124, 3121, 3118, 3114, 3111, 3108, 3105, 3102, 3099, 3096, 3092, 3089,
3086, 3083, 3080, 3077, 3074, 3071, 3067, 3064, 3061, 3058, 3055, 3052, 3049, 3046, 3042, 3039, 3036, 3033, 3030, 3027, 3024,
3021, 3017, 3014, 3011, 3008, 3005, 3002, 2999, 2996, 2992, 2989, 2986, 2983, 2980, 2977, 2974, 2971, 2967, 2964, 2961, 2958,
2955, 2952, 2949, 2945, 2942, 2939, 2936, 2933, 2930, 2927, 2924, 2920, 2917, 2914, 2911, 2908, 2905, 2902, 2899, 2895, 2892,
2889, 2886, 2883, 2880, 2877, 2874, 2870, 2867, 2864, 2861, 2858, 2855, 2852, 2848, 2845, 2842, 2839, 2836, 2833, 2830, 2827,
2823, 2820, 2817, 2814, 2811, 2808, 2805, 2802, 2798, 2795, 2792, 2789, 2786, 2783, 2780, 2776, 2773, 2770, 2767, 2764, 2761,
2758, 2755, 2751, 2748, 2745, 2742, 2739, 2736, 2733, 2730, 2726, 2723, 2720, 2717, 2714, 2711, 2708, 2704, 2701, 2698, 2695,
2692, 2689, 2686, 2683, 2679, 2676, 2673, 2670, 2667, 2664, 2661, 2658, 2654, 2651, 2648, 2645, 2642, 2639, 2636, 2632, 2629,
2626, 2623, 2620, 2617, 2614, 2611, 2607, 2604, 2601, 2598, 2595, 2592, 2589, 2585, 2582, 2579, 2576, 2573, 2570, 2567, 2564,
2560, 2557, 2554, 2551, 2548, 2545, 2542, 2539, 2535, 2532, 2529, 2526, 2523, 2520, 2517, 2513, 2510, 2507, 2504, 2501, 2498,
2495, 2492, 2488, 2485, 2482, 2479, 2476, 2473, 2470, 2466, 2463, 2460, 2457, 2454, 2451, 2448, 2445, 2441, 2438, 2435, 2432,
2429, 2426, 2423, 2419, 2416, 2413, 2410, 2407, 2404, 2401, 2398, 2394, 2391, 2388, 2385, 2382, 2379, 2376, 2372, 2369, 2366,
2363, 2360, 2357, 2354, 2351, 2347, 2344, 2341, 2338, 2335, 2332, 2329, 2325, 2322, 2319, 2316, 2313, 2310, 2307, 2304, 2300,
2297, 2294, 2291, 2288, 2285, 2282, 2278, 2275, 2272, 2269, 2266, 2263, 2260, 2257, 2253, 2250, 2247, 2244, 2241, 2238, 2235,
2231, 2228, 2225, 2222, 2219, 2216, 2213, 2210, 2206, 2203, 2200, 2197, 2194, 2191, 2188, 2184, 2181, 2178, 2175, 2172, 2169,
2166, 2162, 2159, 2156, 2153, 2150, 2147, 2144, 2141, 2137, 2134, 2131, 2128, 2125, 2122, 2119, 2115, 2112, 2109, 2106, 2103,
2100, 2097, 2094, 2090, 2087, 2084, 2081, 2078, 2075, 2072, 2068, 2065, 2062, 2059, 2056, 2053, 2050, 2046, 2043, 2040, 2037,
2034, 2031, 2028, 2025, 2021, 2018, 2015, 2012, 2009, 2006, 2003, 1999, 1996, 1993, 1990, 1987, 1984, 1981, 1978, 1974, 1971,
1968, 1965, 1962, 1959, 1956, 1952, 1949, 1946, 1943, 1940, 1937, 1934, 1930, 1927, 1924, 1921, 1918, 1915, 1912, 1909, 1905,
1902, 1899, 1896, 1893, 1890, 1887, 1883, 1880, 1877, 1874, 1871, 1868, 1865, 1861, 1858, 1855, 1852, 1849, 1846, 1843, 1840,
1836, 1833, 1830, 1827, 1824, 1821, 1818, 1814, 1811, 1808, 1805, 1802, 1799, 1796, 1792, 1789, 1786, 1783, 1780, 1777, 1774,
1770, 1767, 1764, 1761, 1758, 1755, 1752, 1749, 1745, 1742, 1739, 1736, 1733, 1730, 1727, 1723, 1720, 1717, 1714, 1711, 1708,
1705, 1701, 1698, 1695, 1692, 1689, 1686, 1683, 1680, 1676, 1673, 1670, 1667, 1664, 1661, 1658, 1654, 1651, 1648, 1645, 1642,
1639, 1636, 1632, 1629, 1626, 1623, 1620, 1617, 1614, 1610, 1607, 1604, 1601, 1598, 1595, 1592, 1589, 1585, 1582, 1579, 1576,
1573, 1570, 1567, 1563, 1560, 1557, 1554, 1551, 1548, 1545, 1541, 1538, 1535, 1532, 1529, 1526, 1523, 1519, 1516, 1513, 1510,
1507, 1504, 1501, 1498, 1494, 1491, 1488, 1485, 1482, 1479, 1476, 1472, 1469, 1466, 1463, 1460, 1457, 1454, 1450, 1447, 1444,
1441, 1438, 1435, 1432, 1428, 1425, 1422, 1419, 1416, 1413, 1410, 1407, 1403, 1400, 1397, 1394, 1391, 1388, 1385, 1381, 1378,
1375, 1372, 1369, 1366, 1363, 1359, 1356, 1353, 1350, 1347, 1344, 1341, 1337, 1334, 1331, 1328, 1325, 1322, 1319, 1315, 1312,
1309, 1306, 1303, 1300, 1297, 1293, 1290, 1287, 1284, 1281, 1278, 1275, 1272, 1268, 1265, 1262, 1259, 1256, 1253, 1250, 1246,
1243, 1240, 1237, 1234, 1231, 1228, 1224, 1221, 1218, 1215, 1212, 1209, 1206, 1202, 1199, 1196, 1193, 1190, 1187, 1184, 1180,
1177, 1174, 1171, 1168, 1165, 1162, 1159, 1155, 1152, 1149, 1146, 1143, 1140, 1137, 1133, 1130, 1127, 1124, 1121, 1118, 1115,
1111, 1108, 1105, 1102, 1099, 1096, 1093, 1089, 1086, 1083, 1080, 1077, 1074, 1071, 1067, 1064, 1061, 1058, 1055, 1052, 1049,
1045, 1042, 1039, 1036, 1033, 1030, 1027, 1023, 1020, 1017, 1014, 1011, 1008, 1005, 1002, 998, 995, 992, 989, 986, 983, 980, 976,
973, 970, 967, 964, 961, 958, 954, 951, 948, 945, 942, 939, 936, 932, 929, 926, 923, 920, 917, 914, 910, 907, 904, 901, 898, 895, 892,
888, 885, 882, 879, 876, 873, 870, 866, 863, 860, 857, 854, 851, 848, 844, 841, 838, 835, 832, 829, 826, 823, 819, 816, 813, 810, 807,
804, 801, 797, 794, 791, 788, 785, 782, 779, 775, 772, 769, 766, 763, 760, 757, 753, 750, 747, 744, 741, 738, 735, 731, 728, 725, 722,
719, 716, 713, 709, 706, 703, 700, 697, 694, 691, 687, 684, 681, 678, 675, 672, 669, 665, 662, 659, 656, 653, 650, 647, 643, 640, 637,
634, 631, 628, 625, 621, 618, 615, 612, 609, 606, 603, 600, 596, 593, 590, 587, 584, 581, 578, 574, 571, 568, 565, 562, 559, 556, 552,
549, 546, 543, 540, 537, 534, 530, 527, 524, 521, 518, 515, 512, 508, 505, 502, 499, 496, 493, 490, 486, 483, 480, 477, 474, 471, 468,
464, 461, 458, 455, 452, 449, 446, 442, 439, 436, 433, 430, 427, 424, 420, 417, 414, 411, 408, 405, 402, 398, 395, 392, 389, 386, 383,
380, 376, 373, 370, 367, 364, 361, 358, 354, 351, 348, 345, 342, 339, 336, 333, 329, 326, 323, 320, 317, 314, 311, 307, 304, 301, 298,
295, 292, 289, 285, 282, 279, 276, 273, 270, 267, 263, 260, 257, 254, 251, 248, 245, 241, 238, 235, 232, 229, 226, 223, 219, 216, 213,
210, 207, 204, 201, 197, 194, 191, 188, 185, 182, 179, 175, 172, 169, 166, 163, 160, 157, 153, 150, 147, 144, 141, 138, 135, 131, 128,
125, 122, 119, 116, 113, 109, 106, 103, 100, 97, 94, 91, 87, 84, 81, 78, 75, 72, 69, 65, 62, 59, 56, 53, 50, 47, 43, 40, 37, 34, 31, 28, 25,
21, 18, 15, 12, 9, 6, 3};

__prog__ uint8_t __attribute__((space(prog)))
cosLUT_ZeroToPiByTwo_lowByte[0x3FFFu] = {255, 255, 255, 255, 255, 254, 254, 253, 252, 252, 251, 250, 249, 248, 247, 246,
244, 243, 242, 240, 238, 237, 235, 233, 231, 229, 227, 225, 223, 221, 218, 216, 214, 211, 208, 206, 203, 200, 197, 194, 191, 187, 184,
181, 177, 174, 170, 167, 163, 159, 155, 151, 147, 143, 139, 135, 130, 126, 121, 117, 112, 107, 102, 98, 93, 88, 82, 77, 72, 67, 61, 56,
50, 44, 39, 33, 27, 21, 15, 9, 3, 252, 246, 239, 233, 226, 220, 213, 206, 199, 192, 185, 178, 171, 164, 156, 149, 141, 134, 126, 118,
110, 102, 95, 86, 78, 70, 62, 53, 45, 36, 28, 19, 10, 2, 249, 240, 231, 222, 212, 203, 194, 184, 175, 165, 155, 146, 136, 126, 116, 106,
96, 86, 75, 65, 54, 44, 33, 23, 12, 1, 246, 235, 224, 213, 202, 190, 179, 168, 156, 144, 133, 121, 109, 97, 85, 73, 61, 49, 37, 24, 12,
255, 243, 230, 217, 204, 191, 178, 165, 152, 139, 126, 112, 99, 85, 72, 58, 44, 30, 16, 2, 244, 230, 216, 202, 187, 173, 158, 144, 129,
114, 99, 85, 70, 54, 39, 24, 9, 249, 234, 218, 203, 187, 171, 156, 140, 124, 108, 91, 75, 59, 42, 26, 9, 249, 232, 215, 199, 182, 165,

148, 130, 113, 96, 78, 61, 43, 26, 8, 246, 229, 211, 193, 174, 156, 138, 120, 101, 83, 64, 46, 27, 8, 245, 227, 208, 188, 169, 150, 131, 111, 92, 72, 53, 33, 13, 249, 229, 209, 189, 169, 149, 129, 108, 88, 67, 47, 26, 5, 240, 219, 198, 177, 156, 135, 114, 92, 71, 49, 27, 6, 240, 218, 196, 174, 152, 130, 108, 85, 63, 41, 18, 251, 229, 206, 183, 160, 137, 114, 91, 68, 44, 21, 253, 230, 206, 182, 159, 135, 111, 87, 63, 39, 14, 246, 222, 197, 173, 148, 123, 98, 74, 49, 24, 255, 229, 204, 179, 153, 128, 102, 77, 51, 25, 255, 229, 203, 177, 151, 125, 99, 72, 46, 19, 248, 222, 195, 168, 141, 114, 87, 60, 33, 5, 234, 206, 179, 151, 124, 96, 68, 40, 12, 240, 212, 183, 155, 127, 98, 70, 41, 12, 239, 211, 182, 153, 123, 94, 65, 36, 6, 233, 203, 174, 144, 114, 84, 54, 24, 250, 220, 190, 159, 129, 98, 68, 37, 7, 232, 201, 170, 139, 108, 77, 45, 14, 239, 207, 176, 144, 112, 80, 49, 17, 241, 208, 176, 144, 112, 79, 47, 14, 238, 205, 172, 139, 106, 73, 40, 7, 230, 196, 163, 129, 96, 62, 28, 251, 217, 183, 149, 115, 80, 46, 12, 233, 199, 164, 130, 95, 60, 25, 246, 211, 176, 141, 106, 70, 35, 255, 220, 184, 149, 113, 77, 41, 5, 225, 189, 152, 116, 80, 43, 6, 226, 189, 152, 115, 78, 41, 4, 223, 186, 148, 111, 74, 36, 254, 217, 179, 141, 103, 65, 27, 245, 206, 168, 130, 91, 52, 14, 231, 192, 153, 114, 75, 36, 253, 214, 174, 135, 96, 56, 16, 233, 193, 153, 113, 73, 33, 249, 208, 168, 128, 87, 46, 6, 221, 180, 139, 98, 57, 16, 231, 190, 148, 107, 65, 24, 238, 197, 155, 113, 71, 29, 243, 201, 158, 116, 73, 31, 244, 202, 159, 116, 73, 30, 243, 200, 157, 114, 70, 27, 239, 196, 152, 108, 65, 21, 233, 189, 145, 100, 56, 12, 223, 179, 134, 90, 45, 0, 211, 166, 121, 76, 31, 242, 196, 151, 105, 60, 14, 224, 178, 133, 87, 41, 250, 204, 158, 112, 65, 19, 228, 181, 135, 88, 41, 250, 203, 156, 109, 61, 14, 223, 175, 127, 80, 32, 240, 192, 144, 96, 48, 0, 208, 159, 111, 62, 14, 221, 172, 124, 75, 26, 233, 184, 134, 85, 36, 242, 193, 143, 94, 44, 250, 200, 150, 100, 50, 0, 205, 155, 105, 54, 4, 209, 158, 107, 56, 5, 210, 159, 108, 57, 5, 210, 159, 107, 55, 4, 208, 156, 104, 52, 0, 203, 151, 99, 46, 250, 197, 145, 92, 39, 242, 189, 136, 83, 30, 233, 179, 126, 72, 19, 221, 167, 113, 60, 6, 208, 153, 99, 45, 247, 192, 138, 83, 28, 230, 175, 120, 65, 10, 211, 156, 100, 45, 245, 190, 134, 79, 23, 223, 167, 111, 55, 255, 199, 143, 86, 30, 229, 173, 116, 59, 3, 202, 145, 88, 30, 229, 172, 115, 57, 0, 198, 140, 83, 25, 223, 165, 107, 49, 247, 188, 130, 72, 13, 210, 152, 93, 34, 231, 172, 113, 54, 251, 192, 132, 73, 13, 210, 150, 90, 31, 227, 167, 107, 47, 242, 182, 122, 61, 1, 196, 136, 75, 14, 209, 148, 87, 26, 221, 159, 98, 37, 231, 170, 108, 46, 240, 178, 116, 54, 248, 186, 124, 61, 255, 193, 130, 67, 5, 198, 135, 72, 9, 202, 139, 75, 12, 204, 141, 77, 14, 206, 142, 78, 14, 206, 142, 78, 14, 205, 141, 76, 12, 203, 138, 74, 9, 200, 135, 70, 4, 195, 130, 64, 255, 189, 124, 58, 248, 182, 116, 50, 240, 174, 107, 41, 231, 164, 98, 31, 220, 153, 87, 20, 208, 141, 74, 7, 196, 128, 61, 249, 181, 114, 46, 234, 166, 98, 30, 218, 149, 81, 13, 200, 132, 63, 250, 181, 112, 43, 230, 161, 92, 23, 209, 140, 71, 1, 187, 118, 48, 234, 164, 94, 24, 210, 139, 69, 255, 184, 113, 43, 228, 157, 86, 15, 200, 129, 58, 243, 171, 100, 29, 213, 141, 70, 254, 182, 110, 38, 222, 150, 77, 5, 189, 116, 44, 227, 154, 81, 8, 191, 118, 45, 228, 155, 82, 8, 191, 117, 43, 226, 152, 78, 4, 186, 112, 38, 219, 145, 71, 252, 178, 103, 28, 209, 135, 60, 241, 165, 90, 15, 196, 120, 45, 225, 150, 74, 254, 178, 102, 26, 206, 130, 54, 233, 157, 80, 4, 183, 106, 30, 209, 132, 55, 234, 157, 79, 2, 181, 103, 26, 204, 126, 48, 226, 149, 71, 248, 170, 92, 14, 191, 113, 34, 212, 133, 54, 231, 152, 73, 250, 171, 92, 12, 189, 109, 30, 206, 127, 47, 223, 143, 63, 239, 159, 78, 254, 174, 93, 13, 188, 107, 27, 202, 121, 40, 215, 133, 52, 227, 145, 64, 238, 157, 75, 249, 167, 86, 4, 177, 95, 13, 187, 104, 22, 195, 113, 30, 203, 120, 37, 210, 127, 44, 217, 134, 50, 223, 139, 56, 228, 144, 60, 232, 148, 64, 236, 152, 67, 239, 155, 70, 241, 157, 72, 243, 158, 73, 244, 159, 74, 244, 159, 74, 244, 158, 73, 243, 157, 71, 241, 155, 69, 239, 152, 66, 236, 149, 62, 232, 145, 58, 227, 140, 53, 222, 135, 47, 216, 129, 41, 209, 122, 34, 202, 114, 26, 194, 106, 18, 185, 97, 9, 176, 87, 255, 166, 77, 244, 155, 66, 233, 144, 55, 221, 132, 42, 209, 119, 29, 195, 105, 15, 181, 91, 1, 167, 76, 242, 151, 61, 226, 135, 45, 210, 119, 28, 192, 101, 10, 175, 83, 248, 156, 64, 229, 137, 45, 209, 117, 25, 188, 96, 4, 167, 75, 238, 146, 53, 216, 123, 30, 193, 100, 7, 170, 76, 239, 145, 52, 214, 120, 26, 189, 95, 0, 162, 68, 230, 136, 41, 203, 108, 13, 175, 80, 241, 146, 51, 212, 117, 21, 182, 87, 247, 151, 56, 216, 120, 24, 184, 88, 248, 152, 56, 215, 119, 23, 182, 85, 245, 148, 51, 210, 113, 16, 175, 77, 236, 139, 41, 200, 102, 4, 162, 65, 223, 125, 26, 184, 86, 244, 145, 47, 204, 106, 7, 164, 65, 222, 123, 24, 181, 82, 238, 139, 39, 196, 96, 253, 153, 53, 209, 109, 9, 165, 64, 220, 120, 19, 175, 74, 229, 129, 28, 183, 82, 237, 136, 34, 189, 88, 242, 141, 39, 193, 91, 246, 144, 42, 196, 93, 247, 145, 42, 196, 93, 247, 144, 41, 195, 92, 245, 142, 38, 191, 88, 240, 137, 33, 186, 82, 234, 131, 27, 179, 75, 226, 122, 18, 170, 65, 217, 112, 7, 159, 54, 205, 100, 251, 146, 40, 191, 86, 236, 131, 25, 175, 70, 220, 114, 8, 158, 52, 202, 95, 245, 138, 32, 181, 75, 224, 117, 10, 159, 52, 201, 94, 243, 135, 28, 176, 69, 217, 109, 2, 150, 42, 190, 81, 229, 121, 13, 160, 52, 199, 90, 238, 129, 20, 167, 58, 205, 96, 242, 133, 24, 170, 61, 207, 97, 243, 134, 24, 170, 59, 205, 95, 241, 130, 20, 165, 55, 200, 89, 234, 123, 12, 157, 46, 191, 79, 224, 112, 1, 145, 34, 178, 66, 210, 98, 242, 130, 17, 161, 49, 192, 80, 223, 110, 254, 141, 28, 171, 58, 201, 87, 230, 117, 3, 146, 32, 174, 61, 203, 89, 231, 117, 3, 144, 30, 172, 57, 199, 84, 225, 111, 252, 137, 22, 163, 48, 188, 73, 214, 98, 239, 123, 7, 148, 32, 172, 56, 196, 80, 219, 103, 243, 126, 10, 149, 33, 172, 55, 194, 77, 216, 99, 238, 120, 3, 142, 24, 163, 45, 183, 65, 203, 85, 223, 105, 243, 125, 6, 144, 26, 163, 44, 182, 63, 200, 81, 218, 99, 236, 116, 253, 134, 14, 151, 31, 167, 47, 183, 64, 199, 79, 215, 95, 231, 110, 246, 125, 5, 140, 19, 154, 33, 168, 47, 182, 61, 195, 74, 208, 87, 221, 100, 234, 112, 246, 124, 2, 136, 13, 147, 25, 158, 36, 169, 46, 180, 57, 190, 67, 200, 77, 209, 86, 219, 95, 228, 104, 236, 113, 245, 121, 253, 129, 5, 137, 12, 144, 19, 151, 26, 158, 33, 164, 39, 170, 45, 176, 51, 181, 56, 187, 61, 192, 66, 196, 70, 200, 74, 204, 78, 208, 82, 211, 85, 215, 88, 217, 91, 220, 93, 222, 95, 224, 97, 225, 98, 227, 99, 227, 100, 228, 100, 228, 100, 228, 100, 228, 100, 228, 99, 227, 98, 225, 97, 224, 95, 222, 93, 220, 91, 218, 88, 215, 85, 212, 82, 209, 79, 205, 75, 201, 71, 197, 67, 192, 62, 187, 57, 182, 52, 177, 46, 171, 40, 165, 34, 159, 27, 152, 20, 145, 13, 138, 6, 130, 254, 122, 246, 114, 238, 105, 229, 97, 220, 87, 211, 78, 201, 68, 191, 58, 181, 48, 171, 37, 160, 26, 149, 15, 137, 4, 126, 248, 114, 236, 101, 223, 89, 210, 76, 197, 63, 184, 49, 170, 35, 156, 21, 142, 7, 128, 248, 113, 233, 97, 218, 82, 202, 66, 186, 50, 170, 34, 153, 17, 136, 0, 119, 239, 102, 221, 84, 203, 66, 185, 48, 166, 29, 147, 10, 128, 247, 109, 227, 89, 207, 69, 187, 49, 166, 28, 146, 7, 124, 242, 103, 220, 81, 198, 59, 176, 37, 154, 14, 131, 247, 108, 224, 84, 201, 61, 177, 37, 153, 12, 128, 244, 103, 219, 78, 194, 53, 168, 27, 142, 1, 116, 231, 90, 204, 63, 177, 36, 150, 9, 123, 237, 95, 209, 67, 181, 38, 152, 10, 123, 237, 94, 207, 65, 178, 35, 148, 5, 118, 230, 87, 200, 56, 169, 25, 137, 250, 106, 218, 74, 186, 42, 154, 9, 121, 232, 88, 199, 55, 166, 21, 132, 243, 98, 209, 64, 175, 29, 140, 250, 105, 215, 69, 180, 34, 144, 254, 108, 217, 71, 181, 34, 144, 253, 107, 216, 69, 178, 31, 140, 249, 102, 211, 63, 172, 25, 133, 241, 94, 202, 54, 162, 14, 122, 230, 82, 189, 41, 149, 0, 107, 215, 66, 173, 24, 131, 238, 89, 196, 46, 153, 4, 110, 217, 67, 173, 23, 129, 235, 85, 191, 41, 147, 252, 102, 207, 57, 162, 12, 117, 222, 71, 176, 25, 130, 234, 83, 188, 36, 140, 245, 93, 197, 45, 149, 253, 101, 205, 53, 157, 4, 108, 211, 59, 162, 9, 112, 215, 62, 165, 12, 115, 218, 64, 167, 13, 116, 218, 64, 166, 12, 114, 216, 62, 164, 10, 111, 213, 58, 160, 5, 106, 208, 53, 154, 255, 99, 200, 45, 146, 246, 91, 191, 35, 136, 236, 80, 180, 24, 124, 224, 68, 167, 11, 110, 210, 53, 152, 252, 95, 194, 37, 136, 235, 77, 176, 19, 117, 216, 58, 156, 254, 97, 195, 37, 135, 233, 74, 172, 14, 111, 209, 50, 147, 245, 86, 183, 24, 121, 218, 58, 155, 252, 92, 189, 29, 126, 222, 62, 158, 254, 94, 190, 30, 126, 221, 61, 156, 252, 91, 186, 26, 121, 216, 55, 150, 245, 83, 178, 17, 111, 206, 44, 138, 232, 71, 165, 3, 97, 190, 28, 122, 216, 53, 147, 240, 77, 171, 8, 101, 194, 31, 124, 216, 53, 146, 238, 75, 167, 4, 96, 188, 24, 116, 208, 44, 136, 228, 63, 155, 246, 82, 173, 8, 100, 191, 26, 117, 208, 42, 133, 224, 58, 149, 239, 74, 164, 254, 88, 179, 13, 102, 192, 26, 116, 205, 39, 128, 218, 51, 140, 230, 63, 152, 241, 73, 162, 251, 84, 172, 5, 93, 181, 14, 102, 190, 22, 110, 198, 30, 117, 205, 37, 124, 212, 43, 130, 217, 49, 136, 223, 53, 140, 227, 58, 144, 231, 61, 148, 234, 64, 150, 236, 66, 152, 238, 68, 154, 239, 69, 154, 240, 69, 154, 240, 69, 154, 239, 67, 152, 237, 66, 150, 235, 63, 147, 232, 60, 144, 228, 56, 140, 224, 51, 135, 219, 46, 130, 213, 40, 124, 207, 34, 117, 200, 26, 109, 192, 19, 101, 184, 10, 92, 174, 1, 83, 165, 247, 72, 154, 236, 62, 143, 225, 50, 131, 213, 38, 119, 200, 25, 106, 187, 11, 92, 173, 253, 77, 158, 238, 62, 142, 222, 46, 126, 206, 30, 110, 189, 13, 92, 172, 251, 74, 153, 232, 55, 134, 213, 36, 115, 193, 16, 94, 173, 251, 73, 151, 229, 51, 129, 207, 29, 107, 184, 6, 83, 161, 238, 60, 137, 214,

35, 112, 189, 10, 86, 163, 240, 60, 137, 213, 33, 109, 186, 6, 82, 158, 233, 53, 129, 204, 24, 99, 175, 250, 69, 145, 220, 39, 114, 188, 7, 82, 157, 231, 50, 124, 198, 17, 91, 165, 239, 57, 131, 205, 22, 96, 170, 243, 61, 134, 207, 24, 98, 171, 244, 61, 133, 206, 23, 95, 168, 240, 57, 129, 201, 18, 90, 162, 234, 49, 121, 193, 9, 80, 152, 223, 38, 110, 181, 252, 67, 138, 209, 24, 94, 165, 236, 50, 121, 191, 5, 75, 146, 216, 30, 99, 169, 239, 53, 122, 192, 5, 75, 144, 213, 26, 96, 165, 233, 46, 115, 184, 252, 65, 133, 202, 14, 82, 151, 219, 31, 99, 167, 234, 46, 114, 181, 249, 60, 128, 195, 6, 73, 140, 207, 18, 85, 152, 219, 29, 96, 162, 229, 39, 105, 171, 237, 47, 113, 179, 245, 55, 120, 186, 251, 61, 126, 191, 1, 66, 131, 196, 5, 69, 134, 199, 7, 72, 136, 201, 9, 73, 137, 201, 9, 73, 137, 201, 9, 72, 136, 199, 7, 70, 133, 196, 3, 66, 129, 192, 255, 62, 124, 187, 249, 56, 118, 180, 243, 49, 111, 173, 235, 41, 102, 164, 226, 31, 92, 154, 215, 20, 82, 143, 204, 8, 69, 130, 191, 251, 56, 116, 177, 237, 41, 102, 162, 222, 26, 86, 145, 205, 9, 68, 128, 187, 247, 50, 109, 168, 227, 30, 89, 148, 207, 9, 68, 126, 185, 243, 46, 104, 162, 220, 22, 80, 138, 196, 253, 55, 113, 170, 227, 29, 86, 143, 200, 1, 58, 115, 172, 229, 29, 86, 142, 199, 255, 56, 112, 168, 224, 24, 80, 136, 191, 247, 47, 102, 158, 213, 12, 68, 123, 178, 233, 32, 87, 142, 196, 251, 50, 104, 158, 213, 11, 65, 119, 173, 227, 25, 79, 133, 187, 240, 38, 91, 144, 198, 251, 48, 101, 154, 207, 4, 57, 109, 162, 215, 11, 64, 116, 168, 220, 16, 68, 120, 172, 224, 20, 71, 123, 175, 226, 21, 73, 124, 175, 226, 21, 72, 123, 174, 224, 19, 69, 120, 170, 221, 15, 65, 115, 165, 215, 9, 59, 108, 158, 208, 1, 51, 100, 149, 198, 248, 41, 90, 138, 187, 236, 29, 77, 126, 174, 223, 15, 63, 111, 159, 207, 255, 47, 95, 143, 190, 238, 29, 77, 124, 171, 219, 10, 57, 104, 151, 197, 244, 35, 81, 128, 174, 221, 11, 57, 103, 149, 195, 241, 31, 77, 123, 168, 214, 3, 49, 94, 139, 185, 230, 19, 64, 108, 153, 198, 243, 31, 76, 120, 165, 209, 253, 41, 85, 129, 173, 217, 5, 48, 92, 136, 179, 222, 10, 53, 96, 139, 182, 225, 12, 55, 98, 140, 183, 225, 12, 54, 96, 139, 181, 223, 9, 51, 93, 134, 176, 218, 3, 45, 86, 127, 169, 210, 251, 36, 77, 118, 158, 199, 240, 24, 65, 105, 146, 186, 226, 10, 50, 90, 130, 170, 210, 249, 33, 72, 112, 151, 191, 230, 13, 52, 91, 130, 169, 208, 246, 29, 68, 106, 144, 183, 221, 3, 41, 79, 117, 155, 193, 231, 13, 50, 88, 125, 162, 200, 237, 18, 55, 92, 129, 166, 203, 240, 20, 57, 93, 130, 166, 202, 238, 19, 55, 91, 126, 162, 198, 234, 13, 49, 84, 120, 155, 190, 225, 4, 39, 74, 109, 144, 179, 213, 248, 26, 61, 95, 129, 163, 198, 232, 10, 43, 77, 111, 145, 178, 212, 245, 23, 56, 89, 122, 155, 188, 221, 254, 31, 64, 96, 129, 161, 194, 226, 2, 34, 67, 99, 131, 162, 194, 226, 2, 33, 65, 96, 128, 159, 190, 221, 252, 27, 58, 89, 120, 151, 181, 212, 242, 17, 47, 77, 108, 138, 168, 198, 228, 1, 31, 61, 90, 120, 149, 179, 208, 237, 10, 39, 68, 97, 126, 155, 184, 212, 241, 13, 42, 70, 98, 127, 155, 183, 211, 239, 10, 38, 66, 93, 121, 148, 176, 203, 230, 2, 29, 56, 83, 109, 136, 163, 190, 216, 243, 13, 39, 66, 92, 118, 144, 170, 196, 222, 247, 17, 43, 68, 94, 119, 144, 170, 195, 220, 245, 14, 39, 63, 88, 113, 137, 162, 186, 211, 235, 3, 27, 51, 75, 99, 123, 147, 170, 194, 218, 241, 8, 32, 55, 78, 101, 124, 147, 170, 193, 216, 238, 5, 27, 50, 72, 94, 117, 139, 161, 183, 205, 227, 248, 14, 36, 57, 79, 100, 121, 143, 164, 185, 206, 227, 248, 13, 33, 54, 75, 95, 115, 136, 156, 176, 196, 217, 237, 0, 20, 40, 60, 79, 99, 118, 138, 157, 176, 196, 215, 234, 253, 16, 34, 53, 72, 90, 109, 127, 146, 164, 182, 200, 219, 237, 254, 16, 34, 52, 69, 87, 105, 122, 139, 157, 174, 191, 208, 225, 242, 3, 19, 36, 53, 69, 86, 102, 118, 135, 151, 167, 183, 199, 215, 230, 246, 6, 21, 37, 52, 68, 83, 98, 113, 128, 143, 158, 173, 188, 203, 217, 232, 246, 5, 19, 33, 47, 62, 76, 89, 103, 117, 131, 145, 158, 172, 185, 199, 212, 225, 238, 251, 8, 21, 34, 47, 60, 72, 85, 97, 110, 122, 134, 146, 159, 171, 183, 194, 206, 218, 230, 241, 253, 8, 20, 31, 42, 53, 65, 76, 86, 97, 108, 119, 130, 140, 151, 161, 171, 182, 192, 202, 212, 222, 232, 242, 252, 5, 15, 24, 34, 43, 53, 62, 71, 80, 89, 98, 107, 116, 125, 133, 142, 150, 159, 167, 176, 184, 192, 200, 208, 216, 224, 232, 239, 247, 255, 6, 14, 21, 28, 35, 42, 50, 57, 63, 70, 77, 84, 90, 97, 103, 110, 116, 122, 128, 135, 141, 147, 152, 158, 164, 170, 175, 181, 186, 192, 197, 202, 207, 212, 217, 222, 227, 232, 237, 241, 246, 250, 255, 3, 7, 12, 16, 20, 24, 28, 31, 35, 39, 42, 46, 49, 53, 56, 59, 63, 66, 69, 72, 75, 77, 80, 83, 85, 88, 90, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 112, 114, 115, 117, 118, 119, 121, 122, 123, 124, 125, 126, 126, 127, 128, 128, 129, 129, 129, 130, 130, 130, 130, 130, 130, 130, 129, 129, 128, 128, 127, 127, 126, 125, 124, 124, 123, 122, 120, 119, 118, 117, 115, 114, 112, 110, 109, 107, 105, 103, 101, 99, 97, 94, 92, 90, 87, 85, 82, 80, 77, 74, 71, 68, 65, 62, 59, 55, 52, 49, 45, 42, 38, 34, 31, 27, 23, 19, 15, 11, 6, 2, 254, 249, 245, 240, 236, 231, 226, 221, 216, 211, 206, 201, 196, 191, 185, 180, 174, 169, 163, 157, 151, 146, 140, 134, 127, 121, 115, 109, 102, 96, 89, 83, 76, 69, 62, 56, 49, 41, 34, 27, 20, 13, 5, 254, 246, 238, 231, 223, 215, 207, 199, 191, 183, 175, 166, 158, 150, 141, 133, 124, 115, 106, 98, 89, 80, 70, 61, 52, 43, 33, 24, 14, 5, 251, 241, 232, 222, 212, 202, 192, 181, 171, 161, 150, 140, 129, 119, 108, 97, 86, 76, 65, 54, 42, 31, 20, 9, 253, 242, 230, 218, 207, 195, 183, 171, 159, 147, 135, 123, 110, 98, 86, 73, 61, 48, 35, 22, 10, 253, 240, 226, 213, 200, 187, 173, 160, 146, 133, 119, 105, 92, 78, 64, 50, 36, 21, 7, 249, 234, 220, 205, 191, 176, 161, 146, 132, 117, 102, 86, 71, 56, 41, 25, 10, 250, 235, 219, 203, 187, 171, 155, 139, 123, 107, 91, 74, 58, 41, 25, 8, 247, 231, 214, 197, 180, 163, 145, 128, 111, 94, 76, 59, 41, 23, 6, 244, 226, 208, 190, 172, 154, 135, 117, 99, 80, 62, 43, 24, 6, 243, 224, 205, 186, 167, 148, 128, 109, 90, 70, 51, 31, 11, 248, 228, 208, 188, 168, 148, 127, 107, 87, 66, 46, 25, 5, 240, 219, 198, 177, 156, 135, 114, 93, 72, 50, 29, 7, 242, 220, 199, 177, 155, 133, 111, 89, 67, 44, 22, 0, 233, 211, 188, 166, 143, 120, 97, 74, 51, 28, 5, 238, 214, 191, 168, 144, 120, 97, 73, 49, 25, 1, 233, 209, 185, 161, 136, 112, 88, 63, 38, 14, 245, 220, 195, 170, 145, 120, 95, 70, 44, 19, 249, 224, 198, 173, 147, 121, 95, 69, 43, 17, 247, 220, 194, 168, 141, 115, 88, 61, 34, 8, 237, 210, 183, 156, 128, 101, 74, 46, 19, 247, 220, 192, 164, 136, 108, 80, 52, 24, 252, 223, 195, 167, 138, 110, 81, 52, 23, 251, 222, 193, 163, 134, 105, 76, 46, 17, 243, 214, 184, 154, 125, 95, 65, 35, 5, 230, 200, 170, 140, 109, 79, 48, 17, 243, 212, 181, 150, 119, 88, 57, 25, 250, 219, 187, 156, 124, 92, 61, 29, 253, 221, 189, 157, 125, 92, 60, 28, 251, 219, 186, 153, 120, 88, 55, 22, 245, 212, 178, 145, 112, 78, 45, 11, 234, 200, 166, 132, 98, 64, 30, 252, 218, 184, 149, 115, 81, 46, 11, 233, 198, 163, 128, 93, 58, 23, 244, 208, 173, 138, 102, 67, 31, 251, 216, 180, 144, 108, 72, 36, 255, 219, 183, 146, 110, 73, 37, 0, 219, 182, 145, 108, 71, 34, 253, 216, 178, 141, 103, 66, 28, 246, 208, 171, 133, 95, 57, 18, 236, 198, 160, 121, 83, 44, 5, 223, 184, 145, 106, 67, 28, 245, 206, 166, 127, 87, 48, 8, 225, 185, 145, 105, 65, 25, 241, 201, 161, 121, 80, 40, 255, 215, 174, 133, 92, 52, 11, 226, 184, 143, 102, 61, 19, 234, 192, 151, 109, 67, 26, 240, 198, 156, 114, 72, 29, 243, 201, 158, 116, 73, 30, 244, 201, 158, 115, 72, 29, 242, 198, 155, 112, 68, 25, 237, 194, 150, 106, 62, 18, 230, 186, 142, 98, 53, 9, 220, 176, 131, 87, 42, 253, 208, 163, 118, 73, 28, 239, 193, 148, 103, 57, 11, 222, 176, 130, 84, 38, 248, 202, 156, 110, 63, 17, 227, 180, 134, 87, 40, 249, 202, 155, 108, 61, 14, 223, 176, 128, 81, 33, 242, 194, 146, 98, 51, 3, 211, 162, 114, 66, 18, 225, 177, 128, 80, 31, 238, 189, 141, 92, 43, 250, 200, 151, 102, 52, 3, 209, 160, 110, 60, 11, 217, 167, 117, 67, 16, 222, 172, 122, 71, 21, 226, 175, 125, 74, 23, 228, 177, 126, 75, 23, 228, 177, 125, 74, 22, 226, 175, 123, 71, 19, 223, 171, 119, 67, 14, 218, 165, 113, 60, 8, 211, 158, 105, 52, 255, 202, 149, 96, 43, 245, 192, 138, 85, 31, 233, 179, 126, 72, 18, 219, 165, 111, 57, 2, 204, 149, 95, 40, 242, 187, 132, 77, 22, 223, 168, 112, 57, 2, 202, 147, 91, 36, 236, 180, 124, 68, 12, 212, 156, 100, 44, 243, 187, 130, 74, 17, 216, 160, 103, 46, 245, 188, 131, 73, 16, 215, 157, 100, 42, 241, 183, 125, 67, 9, 207, 149, 91, 33, 231, 172, 114, 55, 253, 194, 135, 77, 18, 215, 156, 97, 38, 235, 175, 116, 56, 253, 193, 134, 74, 14, 211, 151, 91, 31, 226, 166, 106, 46, 241, 181, 120, 60, 255, 194, 133, 72, 11, 206, 145, 84, 23, 218, 156, 95, 33, 227, 166, 104, 42, 236, 174, 112, 50, 244, 182, 119, 57, 251, 188, 125, 63, 0, 193, 130, 67, 4, 197, 134, 71, 8, 200, 137, 73, 10, 202, 138, 74, 11, 203, 139, 74, 10, 202, 138, 73, 9, 200, 136, 71, 7, 198, 133, 68, 3, 194, 129, 63, 254, 189, 123, 58, 248, 183, 117, 51, 241, 175, 109, 43, 233, 167, 101, 34, 224, 157, 91, 24, 214, 147, 80, 13, 202, 135, 68, 1, 189, 122, 55, 243, 176, 108, 40, 228, 161, 93, 25, 213, 145, 76, 8, 196, 127, 59, 246, 178, 109, 40, 228, 159, 90, 21, 208, 138, 69, 0, 186, 117, 47, 234, 164, 94, 25, 211, 141, 71, 1, 187, 116, 46, 232, 161, 91, 20, 205, 135, 64, 249, 178, 107, 36, 221, 150, 78, 7, 192, 120, 49, 233, 161, 89, 18, 202, 130, 58, 241, 169, 97, 25, 208, 136, 63, 247, 174, 101, 28, 211, 138, 65, 248, 175, 102, 28, 211, 138, 64, 246, 173, 99, 25, 207, 133, 59, 241, 167, 93, 18, 200, 126, 51, 232, 158, 83, 8, 189, 114, 39, 220, 145, 70, 251, 175, 100, 24, 205, 129, 53, 234, 158, 82, 6, 186, 110, 33, 213, 137, 60, 240, 163, 87, 10, 189, 112, 35, 214, 137, 60, 239, 162, 84, 7, 186, 108, 30, 209, 131, 53, 231,

153, 75, 253, 175, 97, 18, 196, 118, 39, 216, 138, 59, 236, 157, 78, 255, 176, 97, 18, 195, 115, 36, 212, 133, 53, 229, 150, 70, 246, 166, 86, 6, 181, 101, 21, 196, 116, 35, 211, 130, 49, 224, 143, 62, 237, 156, 75, 250, 168, 87, 6, 180, 98, 17, 191, 109, 27, 201, 119, 37, 211, 129, 47, 220, 138, 55, 229, 146, 63, 236, 154, 71, 244, 160, 77, 250, 167, 83, 0, 173, 89, 5, 178, 94, 10, 182, 98, 14, 186, 102, 17, 189, 105, 20, 192, 107, 22, 194, 109, 24, 195, 110, 25, 196, 110, 25, 196, 110, 25, 195, 109, 24, 194, 108, 22, 192, 106, 20, 190, 103, 17, 187, 100, 14, 183, 96, 9, 179, 92, 5, 174, 86, 255, 168, 81, 249, 162, 74, 243, 155, 67, 235, 147, 59, 227, 139, 51, 219, 131, 42, 210, 121, 33, 200, 111, 22, 190, 101, 12, 179, 89, 0, 167, 78, 244, 155, 65, 231, 142, 52, 218, 128, 38, 204, 114, 24, 190, 99, 9, 174, 84, 249, 158, 68, 233, 142, 51, 216, 125, 34, 199, 107, 16, 180, 89, 253, 162, 70, 234, 142, 50, 214, 122, 30, 194, 102, 9, 173, 80, 244, 151, 59, 222, 129, 36, 199, 106, 13, 176, 82, 245, 152, 58, 221, 127, 33, 196, 102, 8, 170, 76, 238, 144, 50, 211, 117, 23, 184, 89, 251, 156, 61, 222, 128, 33, 193, 98, 3, 164, 69, 229, 134, 38, 199, 103, 7, 167, 71, 231, 135, 39, 199, 103, 7, 166, 70, 229, 133, 36, 195, 99, 2, 161, 64, 223, 126, 28, 187, 90, 248, 151, 53, 212, 114, 16, 174, 76, 234, 136, 38, 196, 98, 0, 157, 59, 216, 118, 19, 176, 77, 235, 136, 37, 194, 94, 251, 152, 53, 209, 110, 10, 166, 67, 223, 123, 23, 179, 79, 235, 135, 35, 190, 90, 246, 145, 44, 200, 99, 254, 153, 52, 207, 106, 5, 160, 59, 213, 112, 10, 165, 63, 217, 116, 14, 168, 66, 220, 118, 16, 169, 67, 221, 118, 16, 169, 66, 220, 117, 14, 167, 64, 217, 114, 11, 163, 60, 212, 109, 5, 158, 54, 206, 102, 255, 151, 47, 198, 94, 246, 142, 37, 189, 84, 236, 131, 26, 177, 73, 224, 119, 14, 164, 59, 210, 105, 255, 150, 44, 194, 89, 239, 133, 27, 177, 71, 221, 115, 9, 158, 52, 201, 95, 244, 138, 31, 180, 73, 222, 115, 8, 157, 50, 199, 91, 240, 132, 25, 173, 65, 214, 106, 254, 146, 38, 186, 78, 225, 117, 9, 156, 48, 195, 86, 234, 125, 16, 163, 54, 201, 92, 239, 129, 20, 167, 57, 204, 94, 240, 130, 21, 167, 57, 203, 93, 238, 128, 18, 164, 53, 199, 88, 233, 123, 12, 157, 46, 191, 80, 225, 114, 2, 147, 36, 180, 69, 213, 101, 246, 134, 22, 166, 54, 198, 86, 229, 117, 5, 148, 36, 179, 67, 210, 97, 240, 127, 14, 157, 44, 187, 74, 216, 103, 246, 132, 18, 161, 47, 189, 75, 217, 103, 245, 131, 17, 159, 44, 186, 71, 213, 98, 240, 125, 10, 151, 36, 177, 62, 203, 88, 228, 113, 253, 138, 22, 163, 47, 187, 71, 211, 95, 235, 119, 3, 143, 26, 166, 49, 189, 72, 212, 95, 234, 117, 0, 139, 22, 161, 44, 182, 65, 204, 86, 225, 107, 245, 127, 10, 148, 30, 168, 50, 187, 69, 207, 88, 226, 107, 245, 126, 8, 145, 26, 163, 44, 181, 62, 199, 79, 216, 97, 233, 113, 250, 130, 10, 147, 27, 163, 43, 179, 59, 194, 74, 210, 89, 225, 104, 239, 119, 254, 133, 12, 147, 26, 161, 40, 175, 53, 188, 67, 201, 79, 214, 92, 226, 104, 238, 116, 250, 128, 6, 140, 17, 151, 29, 162, 39, 173, 50, 183, 60, 193, 70, 203, 80, 213, 90, 222, 99, 231, 108, 240, 116, 249, 125, 1, 133, 9, 141, 17, 148, 24, 156, 31, 163, 38, 170, 45, 176, 51, 182, 57, 188, 63, 194, 69, 199, 74, 204, 79, 209, 84, 214, 88, 218, 92, 222, 96, 226, 100, 230, 103, 233, 106, 236, 109, 239, 112, 241, 114, 243, 116, 245, 118, 247, 119, 248, 120, 249, 121, 250, 122, 250, 122, 251, 123, 251, 122, 250, 122, 250, 121, 249, 120, 248, 119, 246, 118, 245, 116, 243, 114, 241, 112, 238, 109, 236, 106, 233, 103, 229, 100, 226, 96, 222, 92, 218, 88, 214, 83, 209, 78, 204, 73, 199, 68, 193, 63, 188, 57, 182, 51, 176, 44, 169, 38, 162, 31, 155, 24, 148, 16, 140, 8, 132, 0, 124, 248, 116, 240, 107, 231, 98, 222, 89, 212, 80, 203, 70, 193, 60, 183, 50, 172, 39, 162, 28, 151, 17, 139, 6, 128, 250, 116, 238, 104, 226, 92, 213, 79, 201, 66, 188, 53, 175, 40, 161, 26, 147, 12, 133, 254, 119, 240, 104, 225, 89, 210, 74, 194, 59, 179, 43, 163, 27, 147, 11, 131, 250, 114, 234, 97, 217, 80, 199, 63, 182, 45, 164, 27, 146, 9, 127, 246, 109, 227, 90, 208, 71, 189, 51, 169, 31, 150, 12, 129, 247, 109, 227, 88, 206, 67, 185, 46, 164, 25, 142, 3, 120, 237, 98, 215, 76, 192, 53, 169, 30, 146, 7, 123, 239, 99, 215, 75, 191, 51, 167, 27, 143, 2, 118, 233, 93, 208, 67, 183, 42, 157, 16, 131, 246, 104, 219, 78, 192, 51, 165, 24, 138, 253, 111, 225, 83, 197, 55, 169, 27, 140, 254, 112, 225, 83, 196, 53, 167, 24, 137, 250, 107, 220, 77, 190, 46, 159, 16, 128, 241, 97, 210, 66, 178, 34, 146, 2, 114, 226, 82, 194, 49, 161, 16, 128, 239, 95, 206, 61, 172, 27, 138, 249, 104, 215, 70, 180, 35, 145, 0, 110, 221, 75, 185, 39, 149, 3, 113, 223, 77, 187, 40, 150, 3, 113, 222, 76, 185, 38, 147, 0, 109, 218, 71, 180, 33, 141, 250, 102, 211, 63, 172, 24, 132, 240, 92, 200, 52, 160, 12, 119, 227, 79, 186, 38, 145, 252, 104, 211, 62, 169, 20, 127, 234, 85, 191, 42, 149, 255, 106, 212, 62, 169, 19, 125, 231, 81, 187, 37, 143, 248, 98, 204, 53, 159, 8, 113, 219, 68, 173, 22, 127, 232, 81, 186, 34, 139, 244, 92, 197, 45, 149, 254, 102, 206, 54, 158, 6, 110, 214, 61, 165, 13, 116, 220, 67, 170, 18, 121, 224, 71, 174, 21, 124, 227, 74, 176, 23, 125, 228, 74, 177, 23, 125, 227, 73, 175, 21, 123, 225, 71, 172, 18, 120, 221, 66, 168, 13, 114, 215, 61, 162, 6, 107, 208, 53, 154, 254, 99, 199, 44, 144, 244, 89, 189, 33, 133, 233, 77, 176, 20, 120, 220, 63, 163, 6, 105, 205, 48, 147, 246, 89, 188, 31, 130, 229, 71, 170, 13, 111, 210, 52, 150, 248, 91, 189, 31, 129, 227, 68, 166, 8, 106, 203, 45, 142, 239, 81, 178, 19, 116, 213, 54, 151, 248, 89, 186, 26, 123, 219, 60, 156, 253, 93, 189, 29, 125, 221, 61, 157, 253, 92, 188, 28, 123, 219, 58, 153, 249, 88, 183, 22, 117, 212, 51, 146, 240, 79, 174, 12, 107, 201, 39, 134, 228, 66, 160, 254, 92, 186, 24, 118, 211, 49, 142, 236, 73, 167, 4, 97, 190, 27, 120, 213, 50, 143, 236, 73, 165, 2, 94, 187, 23, 115, 208, 44, 136, 228, 64, 156, 248, 83, 175, 11, 102, 194, 29, 121, 212, 47, 138, 230, 65, 156, 247, 81, 172, 7, 98, 188, 23, 113, 204, 38, 128, 218, 52, 143, 233, 66, 156, 246, 80, 170, 3, 93, 182, 16, 105, 194, 27, 117, 206, 39, 128, 217, 49, 138, 227, 59, 148, 236, 69, 157, 246, 78, 166, 254, 86, 174, 6, 94, 182, 13, 101, 188, 20, 107, 195, 26, 113, 201, 32, 119, 206, 37, 124, 210, 41, 128, 214, 45, 131, 218, 48, 134, 220, 51, 137, 223, 53, 138, 224, 54, 140, 225, 55, 140, 226, 55, 140, 226, 55, 140, 225, 54, 139, 224, 52, 137, 222, 50, 135, 219, 48, 132, 216, 44, 128, 212, 40, 124, 208, 36, 120, 203, 31, 114, 198, 25, 109, 192, 19, 102, 185, 12, 95, 178, 5, 88, 170, 253, 79, 162, 244, 71, 153, 235, 61, 143, 225, 51, 133, 215, 41, 123, 204, 30, 111, 193, 18, 99, 181, 6, 87, 168, 249, 74, 155, 235, 60, 141, 221, 46, 126, 207, 31, 111, 192, 16, 96, 176, 0, 80, 159, 239, 63, 142, 222, 46, 125, 204, 28, 107, 186, 9, 88, 167, 246, 69, 148, 226, 49, 127, 206, 28, 107, 185, 7, 86, 164, 242, 64, 142, 220, 41, 119, 197, 18, 96, 173, 251, 72, 149, 227, 48, 125, 202, 23, 100, 176, 253, 74, 151, 227, 48, 124, 200, 21, 97, 173, 249, 69, 145, 221, 41, 117, 193, 12, 88, 163, 239, 58, 133, 209, 28, 103, 178, 253, 72, 147, 222, 41, 115, 190, 8, 83, 157, 232, 50, 124, 198, 17, 91, 165, 238, 56, 130, 204, 21, 95, 169, 242, 59, 133, 206, 23, 96, 169, 242, 59, 132, 205, 22, 94, 167, 240, 56, 128, 201, 17, 89, 162, 234, 50, 122, 194, 9, 81, 153, 225, 40, 112, 183, 254, 70, 141, 212, 27, 98, 169, 240, 55, 126, 197, 11, 82, 153, 223, 38, 108, 178, 248, 63, 133, 203, 17, 87, 156, 226, 40, 110, 179, 249, 62, 132, 201, 14, 83, 152, 222, 35, 104, 172, 241, 54, 123, 191, 4, 72, 141, 209, 21, 90, 158, 226, 38, 106, 174, 242, 53, 121, 189, 0, 68, 135, 203, 14, 81, 149, 216, 27, 94, 161, 228, 38, 105, 172, 239, 49, 116, 182, 248, 59, 125, 191, 1, 67, 133, 199, 9, 75, 141, 206, 16, 81, 147, 212, 22, 87, 152, 217, 26, 91, 156, 221, 30, 95, 160, 224, 33, 97, 162, 226, 34, 99, 163, 227, 35, 99, 163, 227, 35, 98, 162, 226, 33, 97, 160, 224, 31, 94, 157, 220, 27, 90, 153, 216, 23, 86, 148, 211, 17, 80, 142, 205, 11, 73, 135, 197, 3, 65, 127, 189, 251, 56, 118, 180, 241, 47, 108, 169, 230, 36, 97, 158, 219, 24, 85, 145, 206, 11, 71, 132, 192, 253, 57, 118, 178, 238, 42, 102, 162, 222, 26, 86, 145, 205, 9, 68, 128, 187, 246, 50, 109, 168, 227, 30, 89, 148, 207, 9, 68, 127, 185, 244, 46, 105, 163, 221, 23, 82, 140, 198, 0, 57, 115, 173, 231, 32, 90, 147, 205, 6, 63, 121, 178, 235, 36, 93, 150, 207, 7, 64, 121, 177, 234, 34, 91, 147, 203, 4, 60, 116, 172, 228, 28, 84, 139, 195, 251, 50, 106, 161, 217, 16, 71, 127, 182, 237, 36, 91, 146, 200, 255, 54, 108, 163, 218, 16, 70, 125, 179, 233, 31, 85, 139, 193, 247, 45, 99, 152, 206, 4, 57, 110, 164, 217, 14, 67, 121, 174, 227, 24, 76, 129, 182, 235, 31, 84, 136, 189, 241, 37, 89, 142, 194, 246, 42, 94, 145, 197, 249, 45, 96, 148, 199, 251, 46, 97, 148, 200, 251, 46, 97, 148, 198, 249, 44, 95, 145, 196, 246, 40, 91, 141, 191, 241, 35, 85, 135, 185, 235, 29, 79, 128, 178, 227, 21, 70, 119, 169, 218, 11, 60, 109, 158, 207, 0, 48, 97, 146, 194, 243, 35, 84, 132, 180, 228, 20, 69, 117, 164, 212, 4, 52, 100, 147, 195, 242, 34, 81, 128, 176, 223, 14, 61, 108, 155, 202, 248, 39, 86, 132, 179, 225, 16, 62, 109, 155, 201, 247, 37, 83, 129, 175, 221, 10, 56, 102, 147, 193, 238, 27, 73, 118, 163, 208, 253, 42, 87, 132, 177, 222, 10, 55, 99, 144, 188, 233, 21, 65, 109, 153, 197, 241, 29, 73, 117, 161, 204, 248, 35, 79, 122, 166, 209, 252, 39, 82, 125, 168, 211, 254, 41, 84, 126, 169, 211, 254, 40, 83, 125, 167, 209, 251, 37, 79, 121, 163, 205, 247, 32, 74, 115, 157, 198, 240, 25, 66, 107, 148, 189, 230, 15, 56, 97, 138, 178, 219, 3, 44, 84, 125, 165, 205, 245, 29, 69, 109, 149, 189, 229, 13, 52, 92, 131, 171, 210, 250, 33, 72, 111, 150, 189, 228, 11, 50, 89, 128, 166, 205, 243, 26, 64, 103, 141, 179, 217, 255, 37, 75, 113, 151, 189, 227, 8, 46,

83, 121, 158, 196, 233, 14, 51, 88, 125, 162, 199, 236, 17, 54, 90, 127, 164, 200, 236, 17, 53, 89, 125, 162, 198, 234, 14, 49, 85, 121, 157, 192, 228, 7, 43, 78, 113, 149, 184, 219, 254, 33, 68, 103, 137, 172, 207, 241, 20, 55, 89, 123, 158, 192, 226, 4, 38, 72, 106, 140, 174, 208, 241, 19, 52, 86, 119, 153, 186, 219, 252, 30, 63, 96, 129, 162, 194, 227, 4, 36, 69, 101, 134, 166, 199, 231, 7, 39, 71, 103, 135, 167, 199, 231, 6, 38, 70, 101, 133, 164, 195, 227, 2, 33, 64, 95, 126, 157, 188, 219, 249, 24, 55, 85, 116, 146, 176, 207, 237, 11, 41, 71, 101, 131, 161, 191, 220, 250, 24, 53, 83, 112, 141, 171, 200, 229, 2, 31, 60, 89, 118, 147, 176, 204, 233, 5, 34, 62, 91, 119, 147, 175, 204, 232, 4, 32, 59, 87, 115, 143, 170, 198, 225, 253, 24, 52, 79, 106, 133, 160, 187, 214, 241, 12, 39, 65, 92, 119, 145, 172, 198, 224, 251, 21, 47, 73, 99, 125, 151, 177, 203, 228, 254, 24, 49, 75, 100, 125, 151, 176, 201, 226, 251, 20, 45, 70, 95, 120, 144, 169, 194, 218, 242, 11, 35, 59, 84, 108, 132, 156, 180, 204, 227, 251, 19, 43, 66, 90, 113, 136, 160, 183, 206, 229, 253, 20, 43, 65, 88, 111, 134, 156, 179, 202, 224, 246, 13, 35, 57, 80, 102, 124, 146, 168, 189, 211, 233, 255, 20, 42, 63, 85, 106, 127, 149, 170, 191, 212, 233, 254, 19, 40, 61, 81, 102, 123, 143, 163, 184, 204, 225, 245, 9, 29, 49, 69, 89, 109, 129, 148, 168, 188, 207, 227, 246, 9, 29, 48, 67, 86, 105, 124, 143, 162, 181, 200, 218, 237, 0, 18, 36, 55, 73, 91, 110, 128, 146, 164, 182, 200, 218, 235, 253, 15, 32, 50, 67, 85, 102, 119, 137, 154, 171, 188, 205, 222, 239, 0, 16, 33, 50, 66, 83, 99, 116, 132, 148, 164, 180, 197, 213, 228, 244, 4, 20, 36, 51, 67, 82, 98, 113, 129, 144, 159, 174, 189, 205, 220, 234, 249, 8, 23, 38, 52, 67, 81, 96, 110, 124, 139, 153, 167, 181, 195, 209, 223, 237, 251, 8, 22, 35, 49, 62, 76, 89, 103, 116, 129, 142, 155, 168, 181, 194, 207, 219, 232, 245, 1, 14, 26, 39, 51, 63, 75, 88, 100, 112, 124, 136, 147, 159, 171, 183, 194, 206, 217, 229, 240, 251, 6, 18, 29, 40, 51, 62, 73, 83, 94, 105, 115, 126, 137, 147, 157, 168, 178, 188, 198, 208, 218, 228, 238, 248, 2, 12, 21, 31, 41, 50, 59, 69, 78, 87, 97, 106, 115, 124, 133, 142, 150, 159, 168, 177, 185, 194, 202, 211, 219, 227, 235, 244, 252, 4, 12, 20, 28, 35, 43, 51, 58, 66, 73, 81, 88, 96, 103, 110, 117, 124, 131, 138, 145, 152, 159, 166, 172, 179, 185, 192, 198, 205, 211, 217, 223, 229, 236, 242, 247, 253, 3, 9, 15, 20, 26, 31, 37, 42, 48, 53, 58, 63, 68, 73, 78, 83, 88, 93, 98, 102, 107, 112, 116, 121, 125, 129, 134, 138, 142, 146, 150, 154, 158, 162, 166, 169, 173, 177, 180, 184, 187, 190, 194, 197, 200, 203, 206, 209, 212, 215, 218, 221, 224, 226, 229, 231, 234, 236, 239, 241, 243, 245, 247, 250, 252, 253, 255, 1, 3, 5, 6, 8, 9, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 26, 27, 27, 27, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 27, 27, 27, 26, 26, 25, 25, 24, 23, 22, 21, 21, 20, 19, 17, 16, 15, 14, 12, 11, 10, 8, 7, 5, 3, 1, 0, 254, 252, 250, 248, 246, 244, 241, 239, 237, 234, 232, 229, 227, 224, 221, 219, 216, 213, 210, 207, 204, 201, 198, 195, 191, 188, 184, 181, 178, 174, 170, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 126, 122, 118, 113, 109, 104, 99, 95, 90, 85, 80, 75, 70, 65, 60, 55, 50, 44, 39, 33, 28, 22, 17, 11, 6, 0, 250, 244, 238, 232, 226, 220, 214, 207, 201, 195, 188, 182, 175, 169, 162, 155, 149, 142, 135, 128, 121, 114, 107, 100, 92, 85, 78, 70, 63, 55, 48, 40, 32, 24, 17, 9, 1, 249, 241, 233, 224, 216, 208, 199, 191, 183, 174, 165, 157, 148, 139, 130, 122, 113, 104, 95, 85, 76, 67, 58, 48, 39, 29, 20, 10, 1, 247, 237, 227, 217, 208, 198, 187, 177, 167, 157, 147, 136, 126, 115, 105, 94, 84, 73, 62, 51, 40, 29, 18, 7, 252, 241, 230, 219, 207, 196, 184, 173, 161, 150, 138, 126, 114, 102, 90, 78, 66, 54, 42, 30, 18, 5, 249, 236, 224, 211, 199, 186, 173, 160, 147, 135, 122, 108, 95, 82, 69, 56, 42, 29, 15, 2, 244, 231, 217, 203, 190, 176, 162, 148, 134, 120, 105, 91, 77, 63, 48, 34, 19, 5, 246, 231, 217, 202, 187, 172, 157, 142, 127, 112, 97, 81, 66, 51, 35, 20, 4, 245, 229, 213, 197, 181, 166, 150, 134, 118, 101, 85, 69, 53, 36, 20, 3, 243, 226, 209, 193, 176, 159, 142, 125, 108, 91, 74, 57, 40, 22, 5, 244, 226, 209, 191, 173, 156, 138, 120, 102, 84, 66, 48, 30, 12, 250, 231, 213, 195, 176, 158, 139, 120, 102, 83, 64, 45, 26, 7, 244, 225, 206, 187, 168, 148, 129, 109, 90, 70, 51, 31, 11, 248, 228, 208, 188, 168, 148, 128, 107, 87, 67, 46, 26, 6, 241, 220, 200, 179, 158, 137, 117, 96, 75, 54, 32, 11, 246, 225, 203, 182, 160, 139, 117, 96, 74, 52, 30, 9, 243, 221, 199, 177, 154, 132, 110, 88, 65, 43, 20, 254, 231, 208, 186, 163, 140, 117, 94, 71, 48, 25, 2, 234, 211, 188, 164, 141, 117, 94, 70, 46, 22, 255, 231, 207, 183, 159, 135, 110, 86, 62, 38, 13, 245, 220, 196, 171, 146, 122, 97, 72, 47, 22, 253, 228, 203, 178, 152, 127, 102, 76, 51, 25, 255, 230, 204, 178, 152, 127, 101, 75, 49, 22, 252, 226, 200, 173, 147, 120, 94, 67, 41, 14, 243, 216, 190, 163, 136, 109, 81, 54, 27, 0, 229, 201, 174, 146, 119, 91, 63, 36, 8, 236, 208, 180, 152, 124, 96, 68, 39, 11, 239, 210, 182, 153, 125, 96, 67, 39, 10, 237, 208, 179, 150, 121, 92, 63, 33, 4, 231, 201, 172, 142, 113, 83, 53, 23, 250, 220, 190, 160, 130, 99, 69, 39, 9, 234, 204, 174, 143, 112, 82, 51, 20, 246, 215, 184, 153, 122, 91, 59, 28, 253, 222, 190, 159, 127, 96, 64, 33, 1, 225, 193, 161, 129, 97, 65, 33, 1, 225, 193, 160, 128, 95, 63, 30, 254, 221, 188, 155, 123, 90, 57, 24, 247, 213, 180, 147, 114, 80, 47, 13, 236, 202, 169, 135, 101, 67, 34, 0, 222, 188, 153, 119, 85, 51, 17, 238, 204, 169, 135, 100, 65, 31, 252, 217, 182, 147, 112, 77, 42, 7, 228, 192, 157, 122, 86, 51, 15, 235, 200, 164, 128, 92, 56, 20, 240, 204, 168, 132, 96, 60, 23, 243, 206, 170, 133, 97, 60, 23, 242, 205, 168, 131, 94, 57, 20, 239, 202, 164, 127, 90, 52, 15, 233, 195, 158, 120, 82, 44, 6, 224, 186, 148, 110, 72, 33, 251, 213, 174, 136, 97, 59, 20, 237, 198, 160, 121, 82, 43, 4, 221, 181, 142, 103, 63, 24, 241, 201, 162, 122, 82, 43, 3, 219, 179, 139, 99, 59, 19, 235, 194, 154, 114, 73, 33, 248, 208, 167, 126, 86, 45, 4, 219, 178, 137, 96, 55, 14, 229, 187, 146, 104, 63, 21, 236, 194, 153, 111, 69, 27, 241, 199, 157, 115, 73, 31, 244, 202, 160, 117, 75, 32, 246, 203, 160, 118, 75, 32, 245, 202, 159, 116, 73, 30, 242, 199, 156, 112, 69, 25, 238, 194, 150, 107, 63, 19, 231, 187, 143, 99, 55, 10, 222, 178, 133, 89, 45, 0, 211, 167, 122, 77, 32, 244, 199, 154, 109, 64, 18, 229, 184, 139, 93, 48, 2, 213, 167, 122, 76, 30, 240, 194, 148, 102, 56, 10, 220, 174, 128, 81, 35, 245, 198, 152, 105, 58, 12, 221, 174, 127, 80, 33, 242, 195, 148, 101, 53, 6, 215, 167, 120, 72, 25, 233, 185, 138, 90, 42, 250, 202, 154, 106, 58, 10, 217, 169, 121, 72, 24, 231, 183, 134, 85, 36, 244, 195, 146, 97, 48, 255, 206, 156, 107, 58, 8, 215, 166, 116, 66, 17, 223, 173, 124, 74, 24, 230, 180, 130, 80, 29, 235, 185, 134, 84, 34, 239, 189, 138, 87, 36, 242, 191, 140, 89, 38, 243, 192, 141, 89, 38, 243, 191, 140, 88, 37, 241, 189, 138, 86, 34, 238, 186, 134, 82, 30, 234, 182, 129, 77, 25, 228, 176, 123, 70, 18, 221, 168, 115, 63, 10, 213, 160, 106, 53, 0, 203, 149, 96, 43, 245, 192, 138, 84, 31, 233, 179, 125, 71, 17, 219, 165, 111, 57, 3, 204, 150, 95, 41, 242, 188, 133, 79, 24, 225, 170, 115, 60, 5, 206, 151, 96, 41, 241, 186, 131, 75, 20, 220, 164, 109, 53, 253, 197, 141, 85, 29, 229, 173, 117, 61, 5, 204, 148, 91, 35, 234, 178, 121, 64, 8, 207, 150, 93, 36, 235, 178, 121, 63, 6, 205, 148, 90, 33, 231, 174, 116, 58, 0, 199, 141, 83, 25, 223, 165, 107, 48, 246, 188, 130, 71, 13, 210, 152, 93, 34, 232, 173, 114, 55, 252, 193, 134, 75, 16, 213, 153, 94, 35, 231, 172, 112, 53, 249, 189, 129, 70, 10, 206, 146, 86, 26, 222, 161, 101, 41, 236, 176, 116, 55, 250, 190, 129, 68, 8, 203, 142, 81, 20, 215, 154, 93, 31, 226, 165, 103, 42, 236, 175, 113, 52, 246, 184, 122, 60, 255, 193, 131, 68, 6, 200, 138, 76, 13, 207, 144, 82, 19, 213, 150, 87, 24, 218, 155, 92, 29, 222, 159, 95, 32, 225, 162, 98, 35, 227, 164, 100, 36, 229, 165, 101, 37, 229, 165, 101, 37, 229, 165, 101, 36, 228, 164, 99, 35, 226, 162, 97, 32, 223, 158, 94, 29, 220, 155, 90, 24, 215, 150, 85, 19, 210, 144, 79, 13, 204, 138, 72, 6, 196, 131, 65, 255, 188, 122, 56, 246, 180, 113, 47, 236, 170, 103, 37, 226, 159, 93, 26, 215, 148, 81, 14, 203, 136, 69, 1, 190, 123, 55, 244, 176, 109, 41, 229, 162, 94, 26, 214, 146, 78, 10, 198, 130, 62, 250, 181, 113, 44, 232, 163, 95, 26, 213, 145, 76, 7, 194, 125, 56, 243, 174, 105, 36, 222, 153, 84, 14, 201, 131, 62, 248, 178, 108, 39, 225, 155, 85, 15, 201, 131, 61, 246, 176, 106, 35, 221, 150, 80, 9, 195, 124, 53, 238, 167, 96, 25, 210, 139, 68, 253, 182, 110, 39, 224, 152, 81, 9, 194, 122, 50, 234, 163, 91, 19, 203, 131, 59, 242, 170, 98, 26, 209, 137, 64, 248, 175, 103, 30, 213, 141, 68, 251, 178, 105, 32, 215, 142, 68, 251, 178, 104, 31, 214, 140, 67, 249, 175, 101, 28, 210, 136, 62, 244, 170, 96, 22, 203, 129, 55, 237, 162, 88, 13, 195, 120, 45, 226, 152, 77, 2, 183, 108, 33, 214, 139, 63, 244, 169, 94, 18, 199, 123, 48, 228, 152, 76, 1, 181, 105, 29, 209, 133, 57, 237, 160, 84, 8, 188, 111, 35, 214, 138, 61, 240, 163, 87, 10, 189, 112, 35, 214, 137, 60, 238, 161, 84, 7, 185, 108, 30, 209, 131, 53, 231, 154, 76, 254, 176, 98, 20, 198, 120, 42, 219, 141, 63, 240, 162, 83, 5, 182, 103, 25, 202, 123, 44, 221, 142, 63, 240, 161, 82, 2, 179, 100, 20, 197, 117, 38, 214, 135, 55, 231, 151, 71, 247, 167, 87, 7, 183, 103, 23, 198, 118, 38, 213, 133, 52, 228, 147, 66, 241, 161, 80, 255, 174, 93, 12, 187, 105, 24, 199, 118, 36, 211, 129, 48, 222, 140, 59, 233, 151, 69, 243, 161, 79, 253, 171, 89, 7, 181, 98, 16, 189, 107, 24, 198, 115, 33, 206, 123, 40, 213, 130, 47, 220, 137, 54, 227, 143, 60, 233, 149, 66, 238, 155, 71, 244, 160, 76, 248, 164, 80, 252, 168, 84, 0, 172, 88, 3, 175,

91, 6, 178, 93, 9, 180, 95, 10, 182, 97, 12, 183, 98, 13, 184, 98, 13, 184, 99, 13, 184, 98, 13, 183, 98, 12, 182, 96, 10, 181, 95, 9, 179, 92, 6, 176, 90, 4, 173, 87, 0, 170, 83, 253, 166, 79, 248, 162, 75, 244, 157, 70, 239, 152, 64, 233, 146, 59, 227, 140, 52, 221, 133, 45, 214, 126, 38, 206, 118, 30, 198, 110, 22, 190, 102, 14, 181, 93, 5, 172, 84, 251, 162, 74, 241, 152, 63, 230, 142, 53, 220, 130, 41, 208, 119, 30, 196, 107, 17, 184, 94, 5, 171, 81, 248, 158, 68, 234, 144, 54, 220, 130, 40, 206, 115, 25, 191, 100, 10, 175, 85, 250, 159, 69, 234, 143, 52, 217, 126, 35, 200, 109, 18, 183, 91, 0, 165, 73, 238, 146, 55, 219, 127, 36, 200, 108, 16, 180, 88, 252, 160, 68, 232, 139, 47, 211, 118, 26, 189, 97, 4, 168, 75, 238, 145, 52, 215, 123, 29, 192, 99, 6, 169, 76, 238, 145, 52, 214, 121, 27, 189, 96, 2, 164, 70, 233, 139, 45, 207, 113, 18, 180, 86, 248, 153, 59, 221, 126, 32, 193, 98, 4, 165, 70, 231, 136, 41, 202, 107, 12, 173, 78, 239, 143, 48, 209, 113, 18, 178, 83, 243, 147, 52, 212, 116, 20, 180, 84, 244, 148, 52, 212, 115, 19, 179, 82, 242, 145, 49, 208, 111, 15, 174, 77, 236, 139, 42, 201, 104, 7, 166, 69, 228, 130, 33, 192, 94, 253, 155, 57, 216, 118, 20, 178, 81, 239, 141, 43, 201, 103, 4, 162, 64, 222, 123, 25, 182, 84, 241, 143, 44, 201, 102, 4, 161, 62, 219, 120, 21, 178, 78, 235, 136, 37, 193, 94, 250, 151, 51, 208, 108, 8, 165, 65, 221, 121, 21, 177, 77, 233, 133, 32, 188, 88, 243, 143, 43, 198, 97, 253, 152, 51, 207, 106, 5, 160, 59, 214, 113, 12, 167, 65, 220, 119, 18, 172, 71, 225, 124, 22, 176, 75, 229, 127, 25, 179, 77, 231, 129, 27, 181, 79, 232, 130, 28, 181, 79, 232, 130, 27, 180, 78, 231, 128, 25, 178, 75, 228, 125, 22, 175, 72, 225, 121, 18, 171, 67, 220, 116, 12, 165, 61, 213, 109, 6, 158, 54, 206, 102, 254, 149, 45, 197, 93, 244, 140, 35, 187, 82, 234, 129, 24, 176, 71, 222, 117, 12, 163, 58, 209, 104, 255, 150, 44, 195, 90, 240, 135, 29, 179, 74, 224, 118, 13, 163, 57, 207, 101, 251, 145, 39, 189, 82, 232, 126, 19, 169, 62, 212, 105, 255, 148, 41, 191, 84, 233, 126, 19, 168, 61, 210, 103, 251, 144, 37, 185, 78, 227, 119, 12, 160, 52, 201, 93, 241, 133, 25, 173, 65, 213, 105, 253, 145, 37, 184, 76, 224, 115, 7, 154, 45, 193, 84, 231, 123, 14, 161, 52, 199, 90, 237, 128, 19, 165, 56, 203, 93, 240, 130, 21, 167, 58, 204, 94, 241, 131, 21, 167, 57, 203, 93, 239, 129, 19, 164, 54, 200, 89, 235, 124, 14, 159, 48, 194, 83, 228, 117, 7, 152, 41, 186, 74, 219, 108, 253, 142, 30, 175, 64, 208, 97, 241, 129, 18, 162, 50, 194, 83, 227, 115, 3, 147, 34, 178, 66, 210, 98, 241, 129, 16, 160, 47, 191, 78, 221, 109, 252, 139, 26, 169, 56, 199, 86, 229, 116, 3, 145, 32, 175, 61, 204, 90, 233, 119, 5, 148, 34, 176, 62, 204, 90, 232, 118, 4, 146, 32, 174, 59, 201, 87, 228, 114, 255, 141, 26, 167, 53, 194, 79, 220, 105, 246, 131, 16, 157, 42, 183, 68, 208, 93, 233, 118, 3, 143, 27, 168, 52, 192, 77, 217, 101, 241, 125, 9, 149, 33, 172, 56, 196, 80, 219, 103, 242, 126, 9, 149, 32, 171, 55, 194, 77, 216, 99, 238, 121, 4, 143, 26, 165, 47, 186, 69, 207, 90, 228, 111, 249, 131, 14, 152, 34, 172, 54, 192, 74, 212, 94, 232, 114, 252, 133, 15, 153, 34, 172, 53, 191, 72, 209, 91, 228, 109, 246, 127, 8, 145, 26, 163, 44, 181, 62, 198, 79, 216, 96, 233, 113, 250, 130, 10, 147, 27, 163, 43, 179, 59, 195, 75, 211, 91, 227, 107, 242, 122, 2, 137, 17, 152, 32, 167, 46, 182, 61, 196, 75, 210, 89, 224, 103, 238, 117, 252, 130, 9, 144, 22, 157, 36, 170, 48, 183, 61, 195, 74, 208, 86, 220, 98, 232, 110, 244, 122, 0, 133, 11, 145, 22, 156, 34, 167, 44, 178, 55, 188, 66, 199, 76, 209, 86, 219, 96, 229, 106, 239, 115, 248, 125, 1, 134, 11, 143, 19, 152, 28, 161, 37, 169, 45, 177, 53, 185, 61, 193, 69, 201, 77, 208, 84, 216, 91, 223, 98, 230, 105, 236, 112, 243, 118, 249, 125, 0, 131, 6, 137, 11, 142, 17, 148, 22, 153, 28, 158, 33, 163, 38, 168, 42, 173, 47, 177, 51, 181, 55, 185, 59, 189, 63, 193, 66, 196, 70, 199, 73, 202, 76, 205, 79, 208, 81, 211, 84, 213, 86, 215, 88, 217, 90, 219, 92, 220, 93, 222, 94, 223, 96, 224, 96, 225, 97, 225, 98, 226, 98, 226, 98, 226, 98, 226, 98, 226, 98, 225, 97, 225, 96, 224, 95, 223, 94, 222, 93, 220, 91, 219, 90, 217, 88, 215, 86, 213, 84, 210, 81, 208, 79, 205, 76, 202, 73, 199, 70, 196, 66, 193, 63, 189, 59, 185, 55, 181, 51, 177, 47, 172, 42, 168, 38, 163, 33, 158, 28, 153, 22, 148, 17, 142, 11, 137, 6, 131, 0, 125, 250, 118, 243, 112, 237, 105, 230, 99, 223, 92, 216, 84, 209, 77, 201, 70, 194, 62, 186, 54, 178, 46, 170, 38, 161, 29, 153, 20, 144, 12, 135, 3, 126, 249, 117, 240, 107, 231, 98, 221, 88, 211, 78, 201, 68, 190, 57, 180, 47, 169, 36, 158, 25, 147, 14, 136, 2, 125, 247, 113, 235, 101, 223, 89, 211, 77, 199, 65, 187, 52, 174, 40, 161, 27, 148, 14, 135, 1, 122, 243, 108, 230, 95, 216, 81, 202, 67, 188, 53, 173, 38, 159, 23, 144, 9, 129, 250, 114, 235, 99, 219, 83, 204, 68, 188, 52, 172, 36, 156, 20, 140, 4, 123, 243, 107, 226, 90, 209, 73, 192, 56, 175, 38, 158, 21, 140, 3, 122, 241, 104, 223, 86, 205, 68, 186, 49, 168, 30, 149, 11, 130, 248, 111, 229, 91, 210, 72, 190, 52, 170, 32, 150, 12, 130, 248, 110, 227, 89, 207, 68, 186, 48, 165, 26, 144, 5, 122, 240, 101, 218, 79, 196, 57, 174, 35, 152, 13, 130, 247, 107, 224, 85, 201, 62, 178, 39, 155, 15, 132, 248, 108, 224, 84, 200, 60, 176, 36, 152, 12, 128, 244, 103, 219, 79, 194, 54, 169, 29, 144, 4, 119, 234, 93, 209, 68, 183, 42, 157, 16, 131, 246, 104, 219, 78, 193, 51, 166, 24, 139, 253, 112, 226, 84, 199, 57, 171, 29, 143, 1, 115, 229, 87, 201, 59, 173, 30, 144, 2, 115, 229, 87, 200, 57, 171, 28, 141, 255, 112, 225, 82, 195, 52, 165, 22, 135, 248, 105, 217, 74, 187, 43, 156, 13, 125, 237, 94, 206, 62, 175, 31, 143, 255, 111, 223, 79, 191, 47, 159, 15, 127, 238, 94, 206, 61, 173, 28, 140, 251, 107, 218, 73, 184, 40, 151, 6, 117, 228, 83, 194, 49, 160, 14, 125, 236, 91, 201, 56, 166, 21, 131, 242, 96, 206, 60, 171, 25, 135, 245, 99, 209, 63, 173, 27, 137, 246, 100, 210, 63, 173, 27, 136, 246, 99, 208, 62, 171, 24, 133, 243, 96, 205, 58, 167, 20, 129, 237, 90, 199, 52, 160, 13, 121, 230, 83, 191, 43, 152, 4, 112, 221, 73, 181, 33, 141, 249, 101, 209, 61, 169, 20, 128, 236, 87, 195, 47, 154, 6, 113, 220, 72, 179, 30, 138, 245, 96, 203, 54, 161, 12, 119, 226, 76, 183, 34, 141, 247, 98, 204, 55, 161, 12, 118, 225, 75, 181, 31, 137, 244, 94, 200, 50, 156, 5, 111, 217, 67, 173, 22, 128, 234, 83, 189, 38, 143, 249, 98, 203, 53, 158, 7, 112, 217, 66, 171, 20, 125, 230, 79, 183, 32, 137, 241, 90, 195, 43, 148, 252, 100, 205, 53, 157, 5, 110, 214, 62, 166, 14, 118, 222, 69, 173, 21, 125, 228, 76, 180, 27, 131, 234, 81, 185, 32, 135, 239, 86, 189, 36, 139, 242, 89, 192, 39, 142, 245, 91, 194, 41, 143, 246, 93, 195, 42, 144, 246, 93, 195, 41, 143, 246, 92, 194, 40, 142, 244, 90, 191, 37, 139, 241, 86, 188, 34, 135, 237, 82, 184, 29, 130, 232, 77, 178, 23, 124, 225, 70, 171, 16, 117, 218, 63, 164, 8, 109, 210, 54, 155, 255, 100, 200, 45, 145, 245, 90, 190, 34, 134, 234, 78, 178, 22, 122, 222, 66, 166, 9, 109, 209, 52, 152, 251, 95, 194, 38, 137, 236, 80, 179, 22, 121, 220, 63, 162, 5, 104, 203, 46, 145, 244, 86, 185, 28, 126, 225, 67, 166, 8, 106, 205, 47, 145, 244, 86, 184, 26, 124, 222, 64, 162, 4, 101, 199, 41, 139, 236, 78, 175, 17, 114, 212, 53, 151, 248, 89, 186, 28, 125, 222, 63, 160, 1, 98, 195, 36, 132, 229, 70, 166, 7, 104, 200, 41, 137, 234, 74, 170, 11, 107, 203, 43, 139, 235, 75, 171, 11, 107, 203, 43, 139, 234, 74, 170, 9, 105, 200, 40, 135, 231, 70, 165, 5, 100, 195, 34, 129, 224, 64, 158, 253, 92, 187, 26, 121, 215, 54, 149, 243, 82, 176, 15, 109, 204, 42, 136, 231, 69, 163, 1, 95, 189, 27, 121, 215, 53, 147, 241, 78, 172, 10, 103, 197, 35, 128, 222, 59, 152, 246, 83, 176, 13, 107, 200, 37, 130, 223, 60, 153, 246, 82, 175, 12, 105, 197, 34, 127, 219, 56, 148, 240, 77, 169, 5, 98, 190, 26, 118, 210, 46, 138, 230, 66, 158, 250, 86, 177, 13, 105, 196, 32, 123, 215, 50, 142, 233, 69, 160, 251, 86, 177, 13, 104, 195, 30, 121, 211, 46, 137, 228, 63, 153, 244, 79, 169, 4, 94, 185, 19, 110, 200, 34, 124, 215, 49, 139, 229, 63, 153, 243, 77, 167, 1, 90, 180, 14, 103, 193, 27, 116, 206, 39, 129, 218, 51, 141, 230, 63, 152, 241, 74, 164, 253, 85, 174, 7, 96, 185, 18, 106, 195, 28, 116, 205, 37, 126, 214, 47, 135, 223, 56, 144, 232, 64, 152, 240, 72, 160, 248, 80, 168, 0, 87, 175, 7, 94, 182, 14, 101, 189, 20, 107, 195, 26, 113, 201, 32, 119, 206, 37, 124, 211, 42, 129, 216, 47, 133, 220, 51, 138, 224, 55, 141, 228, 58, 145, 231, 61, 148, 234, 64, 150, 236, 66, 153, 239, 69, 154, 240, 70, 156, 242, 71, 157, 243, 72, 158, 243, 73, 158, 244, 73, 158, 244, 73, 158, 243, 72, 157, 242, 71, 156, 241, 70, 155, 240, 68, 153, 238, 66, 151, 235, 64, 148, 233, 61, 146, 230, 58, 142, 226, 55, 139, 223, 51, 135, 219, 47, 130, 214, 42, 126, 209, 37, 121, 204, 32, 115, 199, 26, 109, 193, 20, 103, 186, 14, 97, 180, 7, 90, 173, 0, 83, 165, 248, 75, 158, 240, 67, 150, 232, 59, 141, 224, 50, 132, 215, 41, 123, 205, 31, 114, 196, 22, 104, 186, 11, 93, 175, 1, 83, 164, 246, 72, 153, 235, 60, 142, 223, 49, 130, 211, 37, 118, 199, 24, 105, 186, 11, 92, 173, 254, 79, 160, 241, 65, 146, 227, 51, 132, 212, 37, 117, 198, 22, 103, 183, 7, 87, 168, 248, 72, 152, 232, 56, 136, 216, 40, 119, 199, 23, 103, 182, 6, 85, 165, 245, 68, 147, 227, 50, 129, 209, 32, 111, 190, 13, 92, 171, 250, 73, 152, 231, 54, 133, 212, 34, 113, 192, 14, 93, 171, 250, 72, 151, 229, 51, 129, 208, 30, 108, 186, 8, 86, 164, 242, 64, 142, 220, 42, 119, 197, 19, 96, 174, 252, 73, 151, 228, 49, 127, 204, 25, 103, 180, 1, 78, 155, 232, 53, 130, 207, 28, 105, 182, 3, 79, 156, 233, 53, 130, 206, 27, 103, 180, 0, 76, 153, 229, 49, 125, 201, 22, 98, 174, 250, 70, 145, 221, 41, 117, 193, 12, 88, 164, 239, 59, 134, 210, 29, 105, 180, 255, 75, 150, 225, 44, 119, 194, 13, 88, 163, 238, 57, 132, 207, 26,

100, 175, 250, 68, 143, 217, 36, 110, 185, 3, 78, 152, 226, 44, 119, 193, 11, 85, 159, 233, 51, 125, 199, 16, 90, 164, 238, 55, 129, 203, 20, 94, 167, 241, 58, 131, 205, 22, 95, 168, 242, 59, 132, 205, 22, 95, 168, 241, 58, 130, 203, 20, 93, 165, 238, 55, 127, 200, 16, 88, 161, 233, 50, 122, 194, 10, 82, 155, 227, 43, 115, 187, 3, 75, 146, 218, 34, 106, 177, 249, 65, 136, 208, 23, 95, 166, 238, 53, 124, 196, 11, 82, 153, 224, 39, 110, 181, 252, 67, 138, 209, 24, 95, 165, 236, 51, 121, 192, 7, 77, 147, 218, 32, 103, 173, 243, 57, 128, 198, 12, 82, 152, 222, 36, 106, 176, 246, 60, 129, 199, 13, 82, 152, 222, 35, 105, 174, 244, 57, 126, 196, 9, 78, 147, 217, 30, 99, 168, 237, 50, 119, 188, 1, 69, 138, 207, 20, 88, 157, 226, 38, 107, 175, 244, 56, 124, 193, 5, 73, 141, 210, 22, 90, 158, 226, 38, 106, 174, 242, 54, 121, 189, 1, 68, 136, 204, 15, 83, 150, 218, 29, 97, 164, 231, 42, 110, 177, 244, 55, 122, 189, 0, 67, 134, 201, 12, 79, 146, 212, 23, 90, 156, 223, 33, 100, 166, 233, 43, 110, 176, 242, 52, 119, 185, 251, 61, 127, 193, 3, 69, 135, 201, 11, 76, 142, 208, 18, 83, 149, 214, 24, 89, 155, 220, 30, 95, 160, 226, 35, 100, 165, 230, 39, 105, 170, 234, 43, 108, 173, 238, 47, 112, 176, 241, 50, 114, 179, 243, 52, 116, 181, 245, 53, 118, 182, 246, 54, 118, 183, 247, 55, 119, 183, 247, 54, 118, 182, 246, 54, 117, 181, 245, 52, 116, 179, 243, 50, 114, 177, 240, 48, 111, 174, 237, 44, 107, 171, 234, 41, 104, 166, 229, 36, 99, 162, 224, 31, 94, 156, 219, 26, 88, 151, 213, 19, 82, 144, 206, 13, 75, 137, 199, 5, 67, 129, 191, 253, 59, 121, 183, 245, 51, 112, 174, 236, 41, 103, 164, 226, 31, 93, 154, 216, 21, 82, 144, 205, 10, 71, 132, 193, 254, 59, 120, 181, 242, 47, 108, 169, 229, 34, 95, 155, 216, 21, 81, 142, 202, 6, 67, 127, 187, 248, 52, 112, 172, 232, 36, 97, 157, 217, 20, 80, 140, 200, 4, 64, 123, 183, 243, 46, 106, 165, 225, 28, 88, 147, 206, 10, 69, 128, 188, 247, 50, 109, 168, 227, 30, 89, 148, 207, 10, 68, 127, 186, 245, 47, 106, 165, 223, 26, 84, 143, 201, 3, 62, 120, 178, 236, 39, 97, 155, 213, 15, 73, 131, 189, 247, 49, 107, 164, 222, 24, 81, 139, 197, 254, 56, 113, 171, 228, 30, 87, 144, 202, 3, 60, 117, 174, 231, 33, 90, 147, 204, 4, 61, 118, 175, 232, 33, 89, 146, 203, 3, 60, 116, 173, 229, 30, 86, 142, 199, 255, 55, 111, 168, 224, 24, 80, 136, 192, 248, 48, 104, 160, 215, 15, 71, 127, 182, 238, 37, 93, 149, 204, 3, 59, 114, 170, 225, 24, 79, 135, 190, 245, 44, 99, 154, 209, 8, 63, 118, 173, 228, 26, 81, 136, 190, 245, 44, 98, 153, 207, 6, 60, 115, 169, 223, 21, 76, 130, 184, 238, 36, 90, 144, 198, 252, 50, 104, 158, 212, 10, 63, 117, 171, 224, 22, 76, 129, 183, 236, 33, 87, 140, 194, 247, 44, 97, 150, 204, 1, 54, 107, 160, 213, 10, 63, 115, 168, 221, 18, 71, 123, 176, 229, 25, 78, 130, 183, 235, 31, 84, 136, 188, 241, 37, 89, 141, 193, 246, 42, 94, 146, 198, 249, 45, 97, 149, 201, 252, 48, 100, 151, 203, 255, 50, 102, 153, 205, 0, 51, 103, 154, 205, 0, 51, 103, 154, 205, 0, 51, 102, 153, 204, 254, 49, 100, 151, 202, 252, 47, 98, 148, 199, 249, 44, 94, 145, 195, 245, 40, 90, 140, 190, 240, 35, 85, 135, 185, 235, 29, 79, 128, 178, 228, 22, 72, 121, 171, 221, 14, 64, 113, 163, 212, 6, 55, 105, 154, 203, 252, 46, 95, 144, 193, 242, 35, 84, 133, 182, 231, 24, 73, 122, 171, 219, 12, 61, 109, 158, 206, 255, 48, 96, 144, 193, 241, 34, 82, 130, 178, 227, 19, 67, 115, 163, 211, 3, 51, 99, 147, 195, 242, 34, 82, 130, 177, 225, 17, 64, 112, 159, 207, 254, 45, 93, 140, 187, 235, 26, 73, 120, 167, 215, 6, 53, 100, 147, 194, 240, 31, 78, 125, 172, 218, 9, 56, 102, 149, 195, 242, 32, 79, 125, 172, 218, 8, 55, 101, 147, 193, 239, 30, 76, 122, 168, 214, 4, 49, 95, 141, 187, 233, 22, 68, 114, 159, 205, 251, 40, 86, 131, 177, 222, 11, 57, 102, 147, 192, 238, 27, 72, 117, 162, 207, 252, 41, 86, 131, 176, 220, 9, 54, 99, 143, 188, 233, 21, 66, 110, 155, 199, 244, 32, 76, 121, 165, 209, 254, 42, 86, 130, 174, 218, 6, 50, 94, 138, 182, 226, 13, 57, 101, 145, 188, 232, 20, 63, 107, 150, 194, 237, 25, 68, 111, 155, 198, 241, 28, 72, 115, 158, 201, 244, 31, 74, 117, 160, 203, 245, 32, 75, 118, 160, 203, 246, 32, 75, 118, 160, 203, 245, 31, 74, 116, 158, 201, 243, 29, 71, 113, 156, 198, 240, 26, 68, 110, 152, 193, 235, 21, 63, 105, 146, 188, 230, 15, 57, 98, 140, 181, 223, 8, 50, 91, 132, 174, 215, 0, 41, 82, 123, 165, 206, 247, 32, 73, 113, 154, 195, 236, 21, 62, 102, 143, 184, 224, 9, 49, 90, 130, 171, 211, 252, 36, 76, 117, 157, 197, 237, 22, 62, 102, 142, 182, 222, 6, 46, 86, 126, 165, 205, 245, 29, 68, 108, 148, 187, 227, 10, 50, 89, 129, 168, 208, 247, 30, 70, 109, 148, 187, 226, 9, 49, 88, 127, 166, 205, 243, 26, 65, 104, 143, 182, 220, 3, 42, 80, 119, 157, 196, 234, 17, 55, 94, 132, 170, 209, 247, 29, 68, 106, 144, 182, 220, 2, 40, 78, 116, 154, 192, 230, 11, 49, 87, 125, 162, 200, 238, 19, 57, 94, 132, 169, 207, 244, 26, 63, 100, 137, 175, 212, 249, 30, 67, 104, 141, 178, 215, 252, 33, 70, 107, 144, 181, 217, 254, 35, 71, 108, 145, 181, 218, 254, 35, 71, 108, 144, 180, 217, 253, 33, 69, 106, 142, 178, 214, 250, 30, 66, 102, 138, 174, 210, 245, 25, 61, 97, 132, 168, 204, 239, 19, 54, 90, 125, 161, 196, 232, 11, 46, 82, 117, 152, 187, 222, 2, 37, 72, 107, 142, 177, 212, 247, 25, 60, 95, 130, 165, 199, 234, 13, 47, 82, 116, 151, 185, 220, 254, 33, 67, 101, 136, 170, 204, 238, 17, 51, 85, 119, 153, 187, 221, 255, 33, 67, 101, 134, 168, 202, 236, 13, 47, 81, 114, 148, 182, 215, 249, 26, 59, 93, 126, 159, 193, 226, 3, 36, 70, 103, 136, 169, 202, 235, 12, 45, 78, 111, 144, 176, 209, 242, 19, 51, 84, 117, 149, 182, 215, 247, 24, 56, 88, 121, 153, 185, 218, 250, 26, 58, 91, 123, 155, 187, 219, 251, 27, 59, 91, 123, 155, 186, 218, 250, 26, 57, 89, 121, 152, 184, 216, 247, 23, 54, 85, 117, 148, 180, 211, 242, 17, 49, 80, 111, 142, 173, 204, 235, 10, 41, 72, 103, 134, 165, 195, 226, 1, 32, 62, 93, 124, 154, 185, 215, 246, 20, 51, 81, 112, 142, 172, 203, 233, 7, 37, 67, 97, 128, 158, 188, 218, 248, 22, 52, 81, 111, 141, 171, 201, 230, 4, 34, 63, 93, 123, 152, 182, 211, 241, 14, 43, 73, 102, 132, 161, 190, 219, 248, 22, 51, 80, 109, 138, 167, 196, 225, 254, 27, 55, 84, 113, 142, 171, 199, 228, 1, 29, 58, 86, 115, 143, 172, 200, 229, 1, 29, 58, 86, 114, 142, 170, 199, 227, 255, 27, 55, 83, 111, 139, 167, 195, 223, 250, 22, 50, 78, 105, 133, 161, 188, 216, 243, 15, 42, 70, 97, 125, 152, 179, 207, 234, 5, 32, 60, 87, 114, 141, 168, 195, 222, 249, 20, 47, 74, 101, 128, 154, 181, 208, 235, 5, 32, 59, 85, 112, 138, 165, 191, 218, 244, 14, 41, 67, 93, 120, 146, 172, 198, 224, 250, 21, 47, 73, 99, 125, 151, 176, 202, 228, 254, 24, 50, 75, 101, 127, 152, 178, 203, 229, 255, 24, 49, 75, 100, 126, 151, 176, 202, 227, 252, 21, 46, 72, 97, 122, 147, 172, 197, 222, 247, 16, 40, 65, 90, 115, 140, 164, 189, 214, 238, 7, 32, 56, 81, 105, 130, 154, 178, 203, 227, 251, 20, 44, 68, 92, 116, 141, 165, 189, 213, 237, 5, 29, 53, 77, 100, 124, 148, 172, 196, 219, 243, 11, 34, 58, 82, 105, 129, 152, 176, 199, 222, 246, 13, 36, 60, 83, 106, 129, 153, 176, 199, 222, 245, 12, 35, 58, 81, 104, 127, 150, 172, 195, 218, 241, 7, 30, 53, 75, 98, 120, 143, 166, 188, 210, 233, 255, 22, 44, 66, 89, 111, 133, 155, 177, 199, 222, 244, 10, 32, 54, 76, 98, 119, 141, 163, 185, 207, 228, 250, 16, 38, 59, 81, 102, 124, 145, 167, 188, 210, 231, 253, 18, 39, 61, 82, 103, 124, 146, 167, 188, 209, 230, 251, 16, 37, 58, 79, 100, 121, 141, 162, 183, 204, 224, 245, 10, 30, 51, 72, 92, 113, 133, 154, 174, 195, 215, 235, 0, 20, 40, 60, 81, 101, 121, 141, 161, 181, 201, 221, 241, 5, 25, 45, 65, 85, 105, 124, 144, 164, 184, 203, 223, 243, 6, 26, 45, 65, 84, 104, 123, 142, 162, 181, 200, 220, 239, 2, 21, 41, 60, 79, 98, 117, 136, 155, 174, 193, 212, 231, 250, 12, 31, 50, 69, 87, 106, 125, 143, 162, 181, 199, 218, 236, 255, 17, 35, 54, 72, 90, 109, 127, 145, 163, 182, 200, 218, 236, 254, 16, 34, 52, 70, 88, 106, 124, 142, 160, 177, 195, 213, 231, 248, 10, 28, 45, 63, 80, 98, 115, 133, 150, 168, 185, 202, 220, 237, 254, 15, 33, 50, 67, 84, 101, 118, 135, 152, 169, 186, 203, 220, 237, 254, 15, 32, 48, 65, 82, 98, 115, 132, 148, 165, 181, 198, 214, 231, 247, 8, 24, 41, 57, 73, 89, 106, 122, 138, 154, 170, 187, 203, 219, 235, 251, 11, 27, 43, 58, 74, 90, 106, 122, 138, 153, 169, 185, 200, 216, 232, 247, 7, 22, 38, 53, 69, 84, 99, 115, 130, 145, 161, 176, 191, 206, 221, 236, 252, 11, 26, 41, 56, 71, 86, 101, 115, 130, 145, 160, 175, 189, 204, 219, 234, 248, 7, 21, 36, 51, 65, 80, 94, 108, 123, 137, 152, 166, 180, 194, 209, 223, 237, 251, 9, 23, 38, 52, 66, 80, 94, 108, 121, 135, 149, 163, 177, 191, 204, 218, 232, 246, 3, 17, 30, 44, 58, 71, 85, 98, 111, 125, 138, 152, 165, 178, 192, 205, 218, 231, 244, 1, 15, 28, 41, 54, 67, 80, 93, 106, 119, 131, 144, 157, 170, 183, 195, 208, 221, 234, 246, 3, 15, 28, 40, 53, 65, 78, 90, 103, 115, 127, 140, 152, 164, 177, 189, 201, 213, 225, 237, 250, 6, 18, 30, 42, 54, 66, 77, 89, 101, 113, 125, 137, 148, 160, 172, 183, 195, 207, 218, 230, 241, 253, 8, 20, 31, 43, 54, 65, 77, 88, 99, 110, 122, 133, 144, 155, 166, 177, 188, 199, 210, 221, 232, 243, 254, 9, 20, 31, 42, 52, 63, 74, 85, 95, 106, 117, 127, 138, 148, 159, 169, 180, 190, 201, 211, 221, 232, 242, 252, 7, 17, 27, 37, 47, 57, 68, 78, 88, 98, 108, 118, 128, 138, 147, 157, 167, 177, 187, 197, 206, 216, 226, 235, 245, 255, 8, 18, 27, 37, 46, 56, 65, 75, 84, 93, 103, 112, 121, 131, 140, 149, 158, 167, 177, 186, 195, 204, 213, 222, 231, 240, 249, 2, 10, 19, 28, 37, 46, 54, 63, 72, 80, 89, 98, 106, 115, 123, 132, 140, 149, 157, 166, 174, 183, 191, 199, 208, 216, 224, 232, 240, 249, 1, 9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 96, 104, 112, 120, 128, 135, 143, 151, 158, 166, 173, 181, 189, 196, 204, 211, 218, 226, 233, 241, 248, 255, 6, 14, 21, 28, 35, 43, 50, 57, 64, 71, 78, 85, 92, 99, 106, 113, 120, 126,

133, 140, 147, 154, 160, 167, 174, 180, 187, 194, 200, 207, 213, 220, 226, 233, 239, 246, 252, 2, 9, 15, 21, 28, 34, 40, 46, 52, 59, 65, 71, 77, 83, 89, 95, 101, 107, 113, 119, 125, 130, 136, 142, 148, 154, 159, 165, 171, 176, 182, 187, 193, 199, 204, 210, 215, 221, 226, 231, 237, 242, 247, 253, 2, 7, 13, 18, 23, 28, 33, 38, 43, 48, 54, 59, 64, 68, 73, 78, 83, 88, 93, 98, 103, 107, 112, 117, 121, 126, 131, 135, 140, 145, 149, 154, 158, 163, 167, 172, 176, 180, 185, 189, 193, 198, 202, 206, 210, 214, 219, 223, 227, 231, 235, 239, 243, 247, 251, 255, 3, 7, 11, 15, 18, 22, 26, 30, 33, 37, 41, 44, 48, 52, 55, 59, 62, 66, 69, 73, 76, 80, 83, 87, 90, 93, 96, 100, 103, 106, 109, 113, 116, 119, 122, 125, 128, 131, 134, 137, 140, 143, 146, 149, 152, 155, 157, 160, 163, 166, 169, 171, 174, 177, 179, 182, 184, 187, 189, 192, 194, 197, 199, 202, 204, 207, 209, 211, 214, 216, 218, 220, 223, 225, 227, 229, 231, 233, 235, 237, 239, 241, 243, 245, 247, 249, 251, 253, 255, 0, 2, 4, 6, 7, 9, 11, 12, 14, 16, 17, 19, 20, 22, 23, 25, 26, 28, 29, 30, 32, 33, 34, 36, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 54, 55, 56, 57, 58, 58, 59, 60, 60, 61, 62, 62, 63, 63, 64, 65, 65, 65, 66, 66, 67, 67, 67, 68, 68, 68, 69, 69, 69, 69, 69, 69, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 69, 69, 69, 69, 69, 68, 68, 68, 67, 67, 67, 66, 66, 66, 65, 65, 64, 64, 63, 62, 62, 61, 61, 60, 59, 59, 58, 57, 56, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 37, 36, 35, 34, 32, 31, 30, 28, 27, 26, 24, 23, 21, 20, 18, 17, 15, 13, 12, 10, 9, 7, 5, 4, 2, 0, 254, 252, 251, 249, 247, 245, 243, 241, 239, 237, 235, 233, 231, 229, 227, 225, 223, 220, 218, 216, 214, 212, 209, 207, 205, 202, 200, 198, 195, 193, 190, 188, 185, 183, 180, 178, 175, 173, 170, 167, 165, 162, 159, 156, 154, 151, 148, 145, 142, 140, 137, 134, 131, 128, 125, 122, 119, 116, 113, 110, 107, 103, 100, 97, 94, 91, 87, 84, 81, 78, 74, 71, 67, 64, 61, 57, 54, 50, 47, 43, 40, 36, 33, 29, 25, 22, 18, 14, 11, 7, 3, 255, 251, 248, 244, 240, 236, 232, 228, 224, 220, 216, 212, 208, 204, 200, 196, 192, 188, 183, 179, 175, 171, 166, 162, 158, 154, 149, 145, 140, 136, 132, 127, 123, 118, 114, 109, 105, 100, 95, 91, 86, 81, 77, 72, 67, 62, 58, 53, 48, 43, 38, 33, 29, 24, 19, 14, 9, 4, 255, 250, 245, 239, 234, 229, 224, 219, 214, 208, 203, 198, 193, 187, 182, 177, 171, 166, 160, 155, 150, 144, 139, 133, 128, 122, 116, 111, 105, 100, 94, 88, 82, 77, 71, 65, 59, 54, 48, 42, 36, 30, 24, 18, 12, 6, 0, 250, 244, 238, 232, 226, 220, 214, 208, 201, 195, 189, 183, 176, 170, 164, 157, 151, 145, 138, 132, 125, 119, 112, 106, 99, 93, 86, 80, 73, 66, 60, 53, 46, 40, 33, 26, 19, 13, 6, 255, 248, 241, 234, 227, 221, 214, 207, 200, 193, 186, 178, 171, 164, 157, 150, 143, 136, 128, 121, 114, 107, 99, 92, 85, 77, 70, 63, 55, 48, 40, 33, 25, 18, 10, 3, 251, 244, 236, 228, 221, 213, 205, 198, 190, 182, 174, 167, 159, 151, 143, 135, 127, 119, 111, 103, 95, 87, 79, 71, 63, 55, 47, 39, 31, 23, 14, 6, 254, 246, 238, 229, 221, 213, 204, 196, 188, 179, 171, 162, 154, 145, 137, 128, 120, 111, 103, 94, 85, 77, 68, 59, 51, 42, 33, 24, 16, 7, 254, 245, 236, 227, 218, 210, 201, 192, 183, 174, 165, 156, 146, 137, 128, 119, 110, 101, 92, 82, 73, 64, 55, 45, 36, 27, 17, 8, 255, 245, 236, 226, 217, 208, 198, 188, 179, 169, 160, 150, 141, 131, 121, 112, 102, 92, 82, 73, 63, 53, 43, 33, 24, 14, 4, 250, 240, 230, 220, 210, 200, 190, 180, 170, 160, 150, 140, 129, 119, 109, 99, 89, 78, 68, 58, 48, 37, 27, 17, 6, 252, 241, 231, 220, 210, 199, 189, 178, 168, 157, 147, 136, 125, 115, 104, 93, 83, 72, 61, 50, 40, 29, 18, 7, 252, 241, 231, 220, 209, 198, 187, 176, 165, 154, 143, 132, 120, 109, 98, 87, 76, 65, 53, 42, 31, 20, 8, 253, 242, 230, 219, 208, 196, 185, 173, 162, 151, 139, 128, 116, 104, 93, 81, 70, 58, 46, 35, 23, 11, 0, 244, 232, 220, 209, 197, 185, 173, 161, 149, 137, 125, 113, 102, 90, 78, 66, 53, 41, 29, 17, 5, 249, 237, 225, 212, 200, 188, 176, 164, 151, 139, 127, 114, 102, 90, 77, 65, 52, 40, 27, 15, 2, 246, 233, 221, 208, 196, 183, 170, 158, 145, 132, 120, 107, 94, 81, 68, 56, 43, 30, 17, 4, 247, 234, 222, 209, 196, 183, 170, 157, 144, 130, 117, 104, 91, 78, 65, 52, 38, 25, 12, 255, 242, 228, 215, 202, 188, 175, 162, 148, 135, 121, 108, 94, 81, 67, 54, 40, 27, 13, 0, 242, 228, 215, 201, 187, 174, 160, 146, 132, 119, 105, 91, 77, 63, 50, 36, 22, 8, 250, 236, 222, 208, 194, 180, 166, 152, 138, 124, 110, 95, 81, 67, 53, 39, 24, 10, 252, 238, 223, 209, 195, 180, 166, 152, 137, 123, 108, 94, 79, 65, 50, 36, 21, 7, 248, 234, 219, 204, 190, 175, 160, 146, 131, 116, 101, 87, 72, 57, 42, 27, 12, 254, 239, 224, 209, 194, 179, 164, 149, 134, 119, 104, 89, 74, 58, 43, 28, 13, 254, 239, 223, 208, 193, 178, 162, 147, 132, 116, 101, 86, 70, 55, 39, 24, 9, 249, 234, 218, 203, 187, 171, 156, 140, 125, 109, 93, 78, 62, 46, 31, 15, 255, 239, 223, 208, 192, 176, 160, 144, 128, 112, 96, 81, 65, 49, 33, 17, 1, 240, 224, 208, 192, 176, 160, 144, 128, 111, 95, 79, 63, 47, 30, 14, 254, 237, 221, 205, 188, 172, 155, 139, 123, 106, 90, 73, 57, 40, 24, 7, 246, 230, 213, 197, 180, 163, 147, 130, 113, 96, 80, 63, 46, 29, 13, 252, 235, 218, 201, 184, 167, 150, 133, 116, 99, 82, 65, 48, 31, 14, 253, 236, 219, 202, 185, 168, 150, 133, 116, 99, 81, 64, 47, 30, 12, 251, 234, 216, 199, 181, 164, 147, 129, 112, 94, 77, 59, 42, 24, 7, 245, 227, 210, 192, 174, 157, 139, 121, 104, 86, 68, 50, 33, 15, 253, 235, 217, 200, 182, 164, 146, 128, 110, 92, 74, 56, 38, 20, 2, 240, 222, 204, 186, 167, 149, 131, 113, 95, 77, 58, 40, 22, 4, 241, 223, 205, 186, 168, 150, 131, 113, 94, 76, 58, 39, 21, 2, 240, 221, 203, 184, 165, 147, 128, 110, 91, 72, 54, 35, 16, 253, 235, 216, 197, 178, 160, 141, 122, 103, 84, 65, 47, 28, 9, 246, 227, 208, 189, 170, 151, 132, 113, 94, 74, 55, 36, 17, 254, 235, 216, 196, 177, 158, 139, 119, 100, 81, 62, 42, 23, 4, 240, 221, 201, 182, 163, 143, 124, 104, 85, 65, 46, 26, 7, 243, 223, 204, 184, 164, 145, 125, 105, 86, 66, 46, 27, 7, 243, 223, 203, 184, 164, 144, 124, 104, 84, 64, 44, 24, 5, 241, 221, 201, 180, 160, 140, 120, 100, 80, 60, 40, 20, 0, 235, 215, 195, 175, 155, 134, 114, 94, 73, 53, 33, 12, 248, 228, 207, 187, 166, 146, 126, 105, 85, 64, 44, 23, 3, 238, 217, 197, 176, 156, 135, 114, 94, 73, 52, 32, 11, 246, 225, 205, 184, 163, 142, 121, 100, 80, 59, 38, 17, 252, 231, 210, 189, 168, 147, 126, 105, 84, 63, 42, 21, 0, 235, 214, 192, 171, 150, 129, 108, 86, 65, 44, 23, 1, 236, 215, 193, 172, 151, 129, 108, 87, 65, 44, 22, 1, 235, 214, 192, 171, 149, 128, 106, 85, 63, 41, 20, 254, 233, 211, 189, 168, 146, 124, 102, 81, 59, 37, 15, 250, 228, 206, 184, 162, 140, 118, 96, 75, 53, 31, 9, 243, 221, 199, 177, 155, 133, 110, 88, 66, 44, 22, 0, 234, 212, 189, 167, 145, 123, 101, 78, 56, 34, 11, 245, 223, 200, 178, 156, 133, 111, 89, 66, 44, 21, 255, 232, 210, 187, 165, 142, 120, 97, 74, 52, 29, 7, 240, 217, 195, 172, 149, 127, 104, 81, 58, 36, 13, 246, 223, 200, 178, 155, 132, 109, 86, 63, 40, 17, 250, 227, 204, 181, 158, 135, 112, 89, 66, 43, 20, 253, 230, 207, 184, 161, 137, 114, 91, 68, 45, 21, 254, 231, 207, 184, 161, 138, 114, 91, 67, 44, 21, 253, 230, 206, 183, 160, 136, 113, 89, 66, 42, 19, 251, 227, 204, 180, 157, 133, 109, 86, 62, 38, 15, 247, 223, 200, 176, 152, 128, 104, 81, 57, 33, 9, 241, 217, 194, 170, 146, 122, 98, 74, 50, 26, 2, 234, 210, 186, 162, 138, 114, 90, 66, 42, 17, 249, 225, 201, 177, 153, 128, 104, 80, 56, 31, 7, 239, 215, 190, 166, 142, 117, 93, 69, 44, 20, 251, 227, 202, 178, 154, 129, 105, 80, 56, 31, 6, 238, 213, 189, 164, 140, 115, 90, 66, 41, 16, 248, 223, 198, 173, 149, 124, 99, 74, 50, 25, 0, 231, 206, 181, 157, 132, 107, 82, 57, 32, 7, 238, 213, 188, 163, 138, 113, 88, 63, 38, 13, 244, 219, 194, 169, 143, 118, 93, 68, 43, 17, 248, 223, 198, 173, 147, 122, 97, 71, 46, 21, 251, 226, 201, 175, 150, 124, 99, 74, 48, 23, 253, 228, 202, 177, 151, 126, 100, 75, 49, 23, 254, 228, 203, 177, 151, 126, 100, 74, 49, 23, 253, 227, 202, 176, 150, 124, 99, 73, 47, 21, 251, 225, 200, 174, 148, 122, 96, 70, 44, 18, 248, 222, 196, 170, 144, 118, 92, 66, 40, 14, 244, 218, 192, 166, 139, 113, 87, 61, 35, 9, 238, 212, 186, 160, 133, 107, 81, 54, 28, 2, 232, 205, 179, 152, 126, 100, 73, 47, 20, 250, 224, 197, 171, 144, 118, 91, 65, 38, 11, 241, 214, 188, 161, 135, 108, 81, 55, 28, 1, 231, 204, 177, 151, 124, 97, 70, 44, 17, 246, 219, 192, 166, 139, 112, 85, 58, 31, 4, 233, 207, 180, 153, 126, 99, 72, 45, 18, 247, 220, 193, 166, 139, 112, 85, 57, 30, 3, 232, 205, 178, 151, 123, 96, 69, 42, 15, 243, 216, 189, 162, 134, 107, 80, 53, 25, 254, 227, 199, 172, 144, 117, 90, 62, 35, 7, 236, 208, 181, 154, 126, 99, 71, 43, 16, 244, 217, 189, 162, 134, 106, 79, 51, 24, 252, 224, 197, 169, 141, 114, 86, 58, 30, 3, 231, 203, 175, 147, 120, 92, 64, 36, 8, 236, 209, 181, 153, 125, 97, 69, 41, 13, 241, 213, 185, 157, 129, 101, 73, 45, 17, 245, 217, 189, 161, 133, 105, 76, 48, 20, 248, 220, 192, 163, 135, 107, 79, 51, 22, 250, 222, 194, 165, 137, 109, 80, 52, 24, 251, 223, 195, 166, 138, 109, 81, 52, 24, 252, 223, 195, 166, 138, 109, 81, 52, 24, 251, 222, 194, 165, 137, 108, 80, 51, 22, 250, 221, 192, 164, 135, 106, 78, 49, 20, 247, 219, 190, 161, 132, 104, 75, 46, 17, 244, 215, 187, 158, 129, 100, 71, 42, 13, 240, 211, 182, 153, 125, 96, 67, 38, 9, 236, 206, 177, 148, 119, 90, 61, 32, 3, 230, 201, 172, 142, 113, 84, 55, 26, 253, 223, 194, 165, 136, 106, 77, 48, 19, 245, 216, 187, 157, 128, 99, 69, 40, 11, 237, 208, 179, 149, 120, 90, 61, 31, 2, 229, 199, 170, 140, 111, 81, 52, 22, 248, 219, 189, 160, 130, 101, 71, 41, 12, 238, 208, 179, 149, 119, 90, 60, 30, 1, 227, 197, 167, 138, 108, 78, 48, 19, 245, 215, 185, 155, 126, 96, 66, 36, 6, 232, 202, 172, 142, 113, 83, 53, 23, 249, 219, 189, 159, 129, 99, 69, 39, 9, 235, 205, 174, 144, 114, 84, 54, 24, 250,

```
220, 190, 159, 129, 99, 69, 39, 8, 234, 204, 174, 144, 113, 83, 53, 22, 248, 218, 188, 157, 127, 97, 66, 36, 6, 231, 201, 170, 140, 110,
79, 49, 18, 244, 213, 183, 152, 122, 91, 61, 30, 0, 225, 195, 164, 134, 103, 73, 42, 11, 237, 206, 176, 145, 114, 84, 53, 22, 248, 217,
186, 156, 125, 94, 63, 33, 2, 227, 196, 166, 135, 104, 73, 42, 12, 237, 206, 175, 144, 113, 83, 52, 21, 246, 215, 184, 153, 122, 91, 60,
29, 254, 223, 192, 161, 130, 99, 68, 37, 6, 231, 200, 169, 138, 107, 76, 45, 14, 239, 207, 176, 145, 114, 83, 52, 20, 245, 214, 183, 152,
120, 89, 58, 27, 251, 220, 189, 158, 126, 95, 64, 32, 1, 226, 194, 163, 132, 100, 69, 38, 6, 231, 199, 168, 136, 105, 74, 42, 11, 235,
204, 172, 141, 109, 78, 46, 15, 239, 208, 176, 145, 113, 81, 50, 18, 243, 211, 179, 148, 116, 84, 53, 21, 245, 214, 182, 150, 119, 87,
55, 24, 248, 216, 184, 153, 121, 89, 57, 25, 250, 218, 186, 154, 122, 91, 59, 27, 251, 219, 187, 155, 123, 92, 60, 28, 252, 220, 188,
156, 124, 92, 60, 28, 252, 220, 188, 156, 124, 92, 60, 28, 252, 220, 188, 156, 124, 91, 59, 27, 251, 219, 187, 155, 123, 90, 58, 26, 250,
218, 185, 153, 121, 89, 57, 24, 248, 216, 184, 151, 119, 87, 55, 22, 246, 214, 181, 149, 117, 84, 52, 20, 243, 211, 178, 146, 114, 81,
49, 16, 240, 208, 175, 143, 110, 78, 45, 13, 236, 204, 171, 139, 106, 74, 41, 9, 232, 200, 167, 135, 102, 69, 37, 4, 228, 195, 162, 130,
97, 65, 32, 255, 223, 190, 157, 125, 92, 59, 26, 250, 217, 184, 152, 119, 86, 53, 21, 244, 211, 178, 145, 113, 80, 47, 14, 237, 205, 172,
139, 106, 73, 40, 7, 231, 198, 165, 132, 99, 66, 33, 0, 223, 190, 157, 124, 91, 58, 25, 248, 215, 182, 149, 116, 83, 50, 17, 240, 207,
174, 141, 108, 75, 42, 9, 232, 199, 165, 132, 99, 66, 33, 0, 223, 190, 156, 123, 90, 57, 24, 246, 213, 180, 147, 114, 80, 47, 14, 237,
203, 170, 137, 104, 70, 37, 4, 226, 193, 160, 126, 93, 60, 26, 249, 216, 182, 149, 116, 82, 49, 15, 238, 205, 171, 138, 104, 71, 37, 4,
227, 193, 160, 126, 93, 59, 26, 248, 215, 181, 148, 114, 81, 47, 13, 236, 202, 169, 135, 102, 68, 34, 1, 223, 190, 156, 122, 89, 55, 22,
244, 210, 177, 143, 109, 76, 42, 8, 231, 197, 163, 129, 96, 62, 28, 251, 217, 183, 149, 116, 82, 48, 14, 236, 203, 169, 135, 101, 67, 34,
0, 222, 188, 154, 120, 87, 53, 19, 241, 207, 173, 139, 105, 71, 38, 4, 226, 192, 158, 124, 90, 56, 22, 244, 210, 176, 142, 108, 74, 40, 6,
228, 194, 160, 126, 92, 58, 24, 246, 212, 178, 144, 110, 76, 42, 7, 229, 195, 161, 127, 93, 59, 25, 247, 212, 178, 144, 110, 76, 42, 7,
229, 195, 161, 127, 93, 58, 24, 246, 212, 177, 143, 109, 75, 40, 6, 228, 194, 159, 125, 91, 57, 22, 244, 210, 175, 141, 107, 72, 38, 4,
225, 191, 157, 122, 88, 54, 19, 241, 207, 172, 138, 103, 69, 35, 0, 222, 187, 153, 118, 84, 50, 15, 237, 202, 168, 133, 99, 64, 30, 251,
217, 182, 148, 113, 79, 44, 10, 231, 197, 162, 128, 93, 58, 24, 245, 211, 176, 142, 107, 72, 38, 3, 225, 190, 155, 121, 86, 51, 17, 238,
204, 169, 134, 100, 65, 30, 252, 217, 182, 148, 113, 78, 43, 9, 230, 195, 161, 126, 91, 56, 22, 243, 208, 173, 139, 104, 69, 34, 0, 221,
186, 151, 116, 82, 47, 12, 233, 198, 163, 129, 94, 59, 24, 245, 210, 176, 141, 106, 71, 36, 1, 222, 187, 153, 118, 83, 48, 13, 234, 199,
164, 129, 94, 59, 24, 246, 211, 176, 141, 106, 71, 36, 1, 222, 187, 152, 117, 82, 47, 12, 233, 198, 163, 128, 93, 58, 23, 244, 209, 174,
138, 103, 68, 33, 254, 219, 184, 149, 114, 79, 44, 9, 229, 194, 159, 124, 89, 54, 19, 240, 205, 169, 134, 99, 64, 29, 250, 214, 179, 144,
109, 74, 39, 3, 224, 189, 154, 119, 83, 48, 13, 234, 199, 163, 128, 93, 58, 22, 243, 208, 173, 137, 102, 67, 32, 252, 217, 182, 146, 111,
76, 41, 5, 226, 191, 155, 120, 85, 49, 14, 235, 199, 164, 129, 93, 58, 23, 243, 208, 173, 137, 102, 66, 31, 252, 216, 181, 146, 110, 75,
39, 4, 225, 189, 154, 118, 83, 47, 12, 233, 197, 162, 126, 91, 55, 20, 240, 205, 170, 134, 99, 63, 28, 248, 213, 177, 142, 106, 71, 35, 0,
220, 185, 149, 114, 78, 43, 7, 228, 192, 156, 121, 85, 50, 14, 235, 199, 164, 128, 93, 57, 21, 242, 206, 171, 135, 99, 64, 28, 249, 213,
177, 142, 106, 71, 35, 255, 220, 184, 149, 113, 77, 42, 6, 226, 191, 155, 119, 84, 48, 12, 233, 197, 161, 126, 90, 54, 19, 239, 203, 168,
132, 96, 61, 25, 245, 209, 174, 138, 102, 67, 31, 251, 215, 180, 144, 108, 72, 37, 1, 221, 186, 150, 114, 78, 42, 7, 227, 191, 155, 120,
84, 48, 12, 233, 197, 161, 125, 89, 54, 18, 238, 202, 166, 131, 95, 59, 23, 243, 207, 172, 136, 100, 64, 28, 248, 213, 177, 141, 105, 69,
33, 254, 218, 182, 146, 110, 74, 38, 3, 223, 187, 151, 115, 79, 43, 7, 227, 192, 156, 120, 84, 48, 12, 232, 196, 160, 124, 89, 53, 17,
237, 201, 165, 129, 93, 57, 21, 241, 205, 169, 133, 98, 62, 26, 246, 210, 174, 138, 102, 66, 30, 250, 214, 178, 142, 106, 70, 34, 254,
218, 182, 146, 110, 74, 38, 2, 222, 186, 150, 114, 78, 42, 6, 226, 190, 154, 118, 82, 46, 10, 230, 194, 158, 122, 86, 50, 14, 234, 198,
162, 126, 90, 54, 18, 238, 202, 166, 129, 93, 57, 21, 241, 205, 169, 133, 97, 61, 25, 245, 209, 173, 137, 101, 64, 28, 248, 212, 176,
140, 104, 68, 32, 252, 216, 179, 143, 107, 71, 35, 255, 219, 183, 147, 111, 74, 38, 2, 222, 186, 150, 114, 78, 42, 5, 225, 189, 153, 117,
81, 45, 9, 228, 192, 156, 120, 84, 48, 12, 231, 195, 159, 123, 87, 51, 15, 234, 198, 162, 126, 90, 54, 17, 237, 201, 165, 129, 93, 56, 20,
240, 204, 168, 132, 95, 59, 23, 243, 207, 171, 134, 98, 62, 26, 246, 210, 173, 137, 101, 65, 29, 248, 212, 176, 140, 104, 67, 31, 251,
215, 179, 143, 106, 70, 34, 254, 218, 181, 145, 109, 73, 36, 0, 220, 184, 148, 111, 75, 39, 3, 223, 186, 150, 114, 78, 42, 5, 225, 189,
153, 116, 80, 44, 8, 228, 191, 155, 119, 83, 46, 10, 230, 194, 158, 121, 85, 49, 13, 232, 196, 160, 124, 87, 51, 15, 235, 199, 162, 126,
90, 54, 17, 237, 201, 165, 128, 92, 56, 20, 239, 203, 167, 131, 95, 58, 22, 242, 206, 169, 133, 97, 61, 24, 244, 208, 172, 135, 99, 63,
27, 246, 210, 174, 138, 101, 65, 29, 249, 212, 176, 140, 104, 67, 31, 251, 215, 178, 142, 106, 70, 33, 253, 217, 181, 144, 108, 72, 36};

_Q15 _Q15cosPILUT(uint16_t phiOverPI)
{
   switch(phiOverPI)
   {
      case 0x0000: return 0x7FFF;
      case 0x4000:
      case 0xC000: return 0x0000;
      case 0x8000: return 0x8000;
      default:
      {
         if (phiOverPI > 0xC000)
         {
            return cosLUT_ZeroToPiByTwo[-phiOverPI - 1];
         }
         if (phiOverPI > 0x8000)
         {
            return -cosLUT_ZeroToPiByTwo[phiOverPI - 0x8000 - 1];
         }
         if (phiOverPI > 0x4000)
         {
            return -cosLUT_ZeroToPiByTwo[0x8000 - phiOverPI - 1];
         }
         return cosLUT_ZeroToPiByTwo[phiOverPI - 1];
      }

   }
}
```

```
_Q15 _Q15sinPILUT(uint16_t phiOverPI)
{
   switch(phiOverPI)
   {
      case 0x4000: return 0x7FFF;
      case 0x0000:
      case 0x8000: return 0x0000;
      case 0xC000: return 0x8000;
      default:
      {
         if (phiOverPI > 0xC000)
         {
            return -cosLUT_ZeroToPiByTwo[phiOverPI - 0xC000 - 1];
         }
         if (phiOverPI > 0x8000)
         {
            return -cosLUT_ZeroToPiByTwo[0xC000 - phiOverPI - 1];
         }
         if (phiOverPI > 0x4000)
         {
            return cosLUT_ZeroToPiByTwo[phiOverPI - 0x4000 - 1];
         }
         return cosLUT_ZeroToPiByTwo[0x4000 - phiOverPI - 1];
      }

   }
}

_Q31_t _Q23cosPILUT(uint16_t phiOverPI)
{

   _Q31_t retval;

   switch(phiOverPI)
   {
      case 0x0000: retval.Q31 = 0x7FFFFFFF; return retval;
      case 0x4000:
      case 0xC000: retval.Q31 = 0x00000000; return retval;
      case 0x8000: retval.Q31 = 0x80000000; return retval;
      default:
      {
         if (phiOverPI > 0xC000)
         {
            retval.Q23_aligned.Q15 = cosLUT_ZeroToPiByTwo[-phiOverPI - 1];
            retval.Q23_aligned.lowByte = cosLUT_ZeroToPiByTwo_lowByte[-phiOverPI - 1];
            return retval;
         }
         if (phiOverPI > 0x8000)
         {
            retval.Q23_aligned.Q15 = cosLUT_ZeroToPiByTwo[phiOverPI - 0x8000 - 1];
            retval.Q23_aligned.lowByte = cosLUT_ZeroToPiByTwo_lowByte[phiOverPI - 0x8000 - 1];
            retval.Q31 = -retval.Q31;
            return retval;
         }
         if (phiOverPI > 0x4000)
         {
            retval.Q23_aligned.Q15 = cosLUT_ZeroToPiByTwo[0x8000 - phiOverPI - 1];
            retval.Q23_aligned.lowByte = cosLUT_ZeroToPiByTwo_lowByte[0x8000 - phiOverPI - 1];
            retval.Q31 = - retval.Q31;
            return retval;
         }
         retval.Q23_aligned.Q15 = cosLUT_ZeroToPiByTwo[phiOverPI - 1];
         retval.Q23_aligned.lowByte = cosLUT_ZeroToPiByTwo_lowByte[phiOverPI - 1];
         return retval;
      }
   }
}


/*
```

```
 * CS4272config.c
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"
#include "spintronicsConfig.h"
#include "timerDrv.h"

#define CS4272_RST_BAR PORTEbits.RE0

void cs4272Init(void)
{
   // Setup RE1 for connection to RSTbar on CS4272
   ANSELEbits.ANSE0 = 0; //set RE0 to be a digital pin
   TRISEbits.TRISE0 = 0; // set RE0 to be an output

#ifndef SIMULATION_MODE
   CS4272_RST_BAR = 0;
   busy_wait_ms(100); // wait 100ms
   CS4272_RST_BAR = 1;
   busy_wait_ms(1); // wait 2ms
#endif
   i2sInit();//initialize the DCI module for communication with CS4272
}


/*
 * digiPotDrv.c
 *
 * Author: Michael Reinhart Sandstedt
 *
 * First release: September 11th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"
#include "spiTx.h"
#include "constants.h"

float getRBridgeOhms(uint16_t val);
float getRAmpOhms(uint8_t val);
float getU25GainFromU24Ohms(float rg_ohms);
float getU25GainFromU24Code(uint8_t u24_code);
float getU25InverseGainFromU24Ohms(float rg_ohms);
float getU25InverseGainFromU24Code(uint8_t u24_code);
float getU24OhmsFromU25Gain(float u25_gain);
uint8_t getU24CodeFromU25Gain(float u25_gain);

/*
 * void setRBridge(uint16_t val)
 *
 * sets the resisttance of RBridge by controlling U20 and U23 in order to
 * balance the Wheatstone bridge. Units of val are arbitrary
 *
 * See getRBridgeOhms() for a translation algorithm to physical units
 *
 * uint16_t val: value to set RBridge to, in the range of 0 to 1535
 */

void setRBridge(uint16_t val)
{
   uint8_t u20_val;
```

```
    uint8_t u23_val;

    if (val > 1535)
    {
        //maximum value is 1535
        val = 1535;
    }

    /*
     * Bits 8-10 are for U20; grab thes.
     *
     * U20 is an AD5160BRJ25-RL7 and requires an 8-bit code, but we are only not
     * going to use all of the lower bits.  See getRBridgeOhms() for more
     * explanation.  Multiplying bits 8-10 by 51 gives the desired range 0-255
     */
    u20_val = (val >> 8) * 51;
    spiTx(U20, u20_val);

    /*
     * Bits 0-7 are for U23; grab these.
     *
     * U23 is an AD8400ARZI and requires a 10-bit code, where the lower 8 bits
     * correspond to the resistance value and the upper two bits are an address,
     * which is always 0b00 in ths case.
     */
    u23_val = val & 0x00FF;
    spiTx(U23, u23_val);
}


/*
 * void setRAmp(uint8_t val)
 *
 * Set the input resistance for the Wheatstone bridge buffer amplifier.
 * This controls the gain for that amplifier.  Units are arbitrary;
 *
 * See getRAmpOhms() for a translation algorithm to physical units
 *
 * uint8_t val: value to set RBridge to
 */

void setRAmp(uint8_t val)
{
    spiTx(U24, val);
}


/*
 * float getRBridgeOhms(uint16_t val)
 *
 * convert values used internally by the firmware into a resistance of the
 * variable leg of the Wheatstone bridge in units of ohms.
 *
 * U20 (AD5160BRJ25-RL7) and U23 (AD8400ARZI) are connected in series
 * to create this resistance on the pcb
 *
 * uint16_t val: arbitrary units used by the firmware
 *
 * return: Wheatstone bridge resistance in ohms corresonding to val
 */

float getRBridgeOhms(uint16_t val)
{
    uint8_t u20_val;
    uint8_t u23_val;

    if (val > 1535)
    {
        //maximum value is 1535
        val = 1535;
    }
```

```
    u20_val = (val >> 8) * 51;
    u23_val = val & 0x00FF;

    return 110.0 + (  (256 - u20_val) * 5000.0
                + (256 - u23_val) * 1000.0 ) / 256.0;
}


/*
 * float getRAmpOhms(uint8_t val)
 *
 * convert values used internally by the firmware into the feedback resistance
 * for the Wheatstone bridge buffer amplifier in units of ohms.
 *
 * uint8_t val: arbitrary units used by the firmware
 *
 * return: Wheatstone bridge buffer amp r_feedback in ohms
 */

inline float getRAmpOhms(uint8_t val)
{
    /*
     * val corresponds to values sent to U24 (AD8400ARZI).  See the Analog
     * Devices datasheet for more info.
     */

    return 50.0 + (256 - val) * 1000.0 / 256.0;
}

inline float getU25GainFromU24Ohms(float rg_ohms)
{
    return 1 + 6000.0 / rg_ohms;
}

inline float getU25GainFromU24Code(uint8_t u24_code)
{
    float rg_ohms;
    rg_ohms = getRAmpOhms(u24_code);
    return getU25GainFromU24Ohms(rg_ohms);
}

inline float getBridgeBufGainFromU24Code(uint8_t u24_code)
{
    return getU25GainFromU24Code(u24_code) * U2_BUF_GAIN;
}

inline float getU25InverseGainFromU24Ohms(float rg_ohms)
{
    return rg_ohms / (6000.0 + rg_ohms);
}

inline float getU25InverseGainFromU24Code(uint8_t u24_code)
{
    float rg_ohms;
    rg_ohms = getRAmpOhms(u24_code);
    return getU25InverseGainFromU24Ohms(rg_ohms);
}

inline float getBridgeInverseGainFromU24Code(uint8_t u24_code)
{
    return getU25InverseGainFromU24Code(u24_code) * U2_INVERSE_GAIN;
}

inline float getU24OhmsFromU25Gain(float u25_gain)
{
    return 6000.0 / (u25_gain - 1);
}

inline uint8_t getU24CodeFromU25Gain(float u25_gain)
{
```

```
    uint8_t u24_code;
    float u24_float_code;
    float u24_ohms;

    u24_ohms = getU24OhmsFromU25Gain(u25_gain);

    u24_float_code = 256 - 256 * (u24_ohms - 50.0) * 1e-3;

    if (u24_float_code < 0.0) {
        u24_code = 0;
    } else if (u24_float_code > 255.0) {
        u24_code = 0xFF;
    } else {
        u24_code = u24_float_code;
    }

    return u24_code;

}

inline uint8_t getU24CodeFromBrdigeBufGain(float bridge_gain) {

    return getU24CodeFromU25Gain(bridge_gain * U2_INVERSE_GAIN);

}


/*
 * File:   asmFIR.c
 * Author: Michael Sandstedt
 *
 * Created on December 2, 2013, 7:02 PM
 */

#include "spintronicsIncludes.h"
#include "spintronicsConfig.h"
#include "spintronicsStructs.h"

#include "AAF/input_BPF.h"
#include "FIR.h"

#define STG12_DECIMATION_FACTOR 32;

inline _Q31_t
stg3_mac(register _Q31_t acc, register _Q31_t coef, register _Q31_t stg3_input);

inline _Q31_t
aaf_mpy(register _Q31_t *dly_line, register uint8_t *dly_line_end_byte,
        register _Q31_t *newest_sample, const _Q15 *coef,
        register uint16_t tapsBy2Less2);

static int16_t delayLine[INPUT_BPF_TAPS] __attribute__((space(xmemory), eds, aligned));

inline int16_t FIR(register int16_t input)
{
    static int16_t *dly_ptr = delayLine;
    volatile register int16_t *reg_dly_ptr asm("w9");
    volatile register int16_t *reg_coef_ptr asm("w11");
    register uint16_t en_modulo = 0x8009;
    volatile register int16_t result;

    reg_dly_ptr = dly_ptr;
    reg_coef_ptr = (int16_t *)input_BPF;

#ifdef BPF_IS_EVEN_ORDER
    __asm__ volatile ("MOV %4, MODCON\n" // enable modulo addressing for w9
                      "MOV %2, [%1]\n" // copy the new sample into the delay line
                      "CLR A, [%1]-=2, w4, [%3]+=2, w5\n" // clr A, prefetch w4, w5, postdec w9, postinc w11
                      "DO "DO_CONST", ACCUMULATE\n" // ulit must be (INPUT_BPF_TAPS - 3) / 2 - 1
                      "MAC w4*w5, A, [%1]-=2, w4, [%3]+=2, w5\n" // MAC, prefetch w4/w5, postdec w9, postinc w11
                      "ACCUMULATE: MAC w4*w5, A, [%1]-=2, w4, [%3]+=2, w5\n" // MAC, prefetch w4/w5, postdec w9, postinc w11
```

```
                "MAC w4*w5, A, [%1]-=2, w4, [%3]+=2, w5\n" // MAC, prefetch w4/w5, postdec w9, postinc w11
                "MAC w4*w5, A, [%1], w4, [%3], w5\n" // MAC, prefetch w4/w5
                "MAC w4*w5, A\n" // MAC
                "CLR MODCON\n" // disable modulo addressing
                "SAC.R A, %0" // return result
                : "=r"(result), "+x"(reg_dly_ptr)
                : "r"(input), "y"(reg_coef_ptr), "r"(en_modulo)
                : "w4", "w5");
#else
    __asm__ volatile ("MOV %4, MODCON\n" // enable modulo addressing for w9
                "MOV %2, [%1]\n" // copy the new sample into the delay line
                "CLR A, [%1]-=2, w4, [%3]+=2, w5\n" // clr A, prefetch w4, w5, postdec w9, postinc w11
                "DO "DO_CONST", ACCUMULATE\n" // ulit must be (INPUT_BPF_TAPS - 2) / 2 - 1
                "MAC w4*w5, A, [%1]-=2, w4, [%3]+=2, w5\n" // MAC, prefetch w4/w5, postdec w9, postinc w11
                "ACCUMULATE: MAC w4*w5, A, [%1]-=2, w4, [%3]+=2, w5\n" // MAC, prefetch w4/w5, postdec w9, postinc w11
                "MAC w4*w5, A, [%1], w4, [%3], w5\n" // MAC, prefetch w4/w5
                "MAC w4*w5, A\n" // MAC
                "CLR MODCON\n" // disable modulo addressing
                "MOV ACCAH, %0" // return result
                : "=r"(result), "+x"(reg_dly_ptr)
                : "r"(input), "y"(reg_coef_ptr), "r"(en_modulo)
                : "w4", "w5");
#endif
    dly_ptr = (int16_t *)reg_dly_ptr;

#ifdef PROBE_POST_FIR
    TXBUF2 = result;
    TXBUF3 = 0;
#endif

    return result;

}

void firInit(void)
{
    XMODSRT = (uint16_t)delayLine;
    XMODEND = (uint16_t)((uint8_t *)delayLine + INPUT_BPF_TAPS * 2 - 1);
    CORCONbits.SATA = 1;
    CORCONbits.ACCSAT = 1;
}

inline _Q31_t
stg3_mac(register _Q31_t acc, register _Q31_t coef, register _Q31_t stg3_input)
{
    register _Q31_t result;
    __asm__ volatile (/* CLR accumulator A */
                "CLR A\n"
                /*
                 * shift coef low-word right by 1 to make sure it is
                 * interpreted as positive
                 */
                "LSR %2, #1, w4\n"
                /* copy input high-word into w5 */
                "MOV %d3, w5\n"
                /* MAC coef-low x input-high */
                "MAC w4*w5, A\n"
                /* copy coef high-word into w4 */
                "MOV %d2, w4\n"
                /*
                 * shift input low-word right by 1 to make sure it is
                 * interpreted as positive
                 */
                "LSR %3, #1, w5\n"
                /* MAC coef-high x input-low */
                "MAC w4*w5, A\n"
                /*
                 * low words already right-shifted by 1; shift right by
                 * 15 more bits to align for Q31 format
                 */
                "SFTAC A, #15\n"
```

```
                /* copy input high-word into w5 */
                "MOV %d3, w5\n"
                /* MAC coef-high x input-high */
                "MAC w4*w5, A\n"
                /* load the previous accumulated value in B */
                "LAC %d1, B\n"
                "MOV %1, ACCBL\n"
                /* add the accumulators and place the result in A */
                "ADD A\n"
                /* return low-word */
                "MOV ACCAL, %0\n"
                /* return high-word */
                "MOV ACCAH, %d0\n"
                : "=r"(result)
                : "r"(acc), "r"(coef), "r"(stg3_input)
                : "w4", "w5");

   return result;

}

inline _Q31_t
aaf_mpy(register _Q31_t *dly_line, register uint8_t *dly_line_end_byte,
      register _Q31_t *newest_sample, const _Q15 *coef,
      register uint16_t tapsBy2Less2)
{
   register uint16_t en_modulo = 0x40B0;
   volatile register _Q31_t result;

   __asm__ volatile (/* set YMODSRT to point to the first byte in the delay */
                "MOV %3, YMODSRT\n"
                /* set YMODEND to point to the last byte in the delay */
                "MOV %4, YMODEND\n"
                /* MODCON = 0x40B0, enable modulo for wll and w9 */
                "MOV %5, MODCON\n"
                /* set w9 to point at the first coefficient */
                "MOV %2, w9\n"
                /* set w11 to point at the first sample */
                "MOV %6, w11\n"
                /*
                 * clear ACCA, prefetch w5 and w4, postinc w9 by 2 to
                 * point to the next coefficient, postdec w11 by 4 to
                 * point to the previous sample in the dly line
                 */
                "CLR A, [w9]+=2, w5, [w11]-=4, w4\n"
                /* must be INPUT_BPF_TAPS / 2 - 2 */
                "DO %1, LPF_ACCUM_LOW\n"
                /*
                 * low-word taps 0 to INPUT_BPF_TAPS - 4; MAC, prefetch w5 &
                 * w4, postinc w9 by 2 to point to the next coefficient,
                 * postdec w11 by 4 to point to the previous sample
                 */
                "MAC w5*w4, A, [w9]+=2, w5, [w11]-=4, w4\n"
                /*
                 * low-word taps 1 to INPUT_BPF_TAPS - 3; MAC, prefetch w5 &
                 * w4, postinc w9 by 2 to point to the next coefficient,
                 * postdec w11 by 4 to point to the previous sample
                 */
                "LPF_ACCUM_LOW: MAC w5*w4, A, [w9]+=2, w5, [w11]-=4, w4\n"
                /*
                 * low-word tap INPUT_BPF_TAPS - 2; MAC, prefetch w5 & w4,
                 * postinc w9 by 2 to point to the next coefficient
                 */
                "MAC w5*w4, A, [w9], w5, [w11], w4\n"
                /* set w9 to point at the first coefficient */
                "MOV %2, w9\n"
                /* set w11 to point at the first sample */
                "MOV %6, w11\n"
                /* increment w11 by 2 to point at the high word*/
                "ADD w11, #2, w11\n"
                /*
```

```
                 * low-word tap INPUT_BPF_TAPS - 1; MAC, prefetch w5 & w4,
                 * postinc w9 by 2 to point to the next coefficient,
                 * postdec w11 by 4 to point to the previous sample's
                 * high word
                 */
                "MAC w5*w4, A, [w9]+=2, w5, [w11]-=4, w4\n"
                /*
                 * low words already right-shifted by 1; shift right by
                 * 15 more bits to align for Q31 format
                 */
                "SFTAC A, #15\n"
                /* must be INPUT_BPF_TAPS / 2 - 2 */
                "DO %1, LPF_ACCUM_HIGH\n"
                /*
                 * high-word taps 0 to INPUT_BPF_TAPS - 4; MAC, prefetch w5 &
                 * w4, postinc w9 by 2 to point to the next coefficient,
                 * postdec w11 by 4 to point to the previous sample
                 */
                "MAC w5*w4, A, [w9]+=2, w5, [w11]-=4, w4\n"
                /*
                 * high-word taps 1 to INPUT_BPF_TAPS - 3; MAC, prefetch w5 &
                 * w4, postinc w9 by 2 to point to the next coefficient,
                 * postdec w11 by 4 to point to the previous sample
                 */
                "LPF_ACCUM_HIGH: MAC w5*w4, A, [w9]+=2, w5, [w11]-=4, w4\n"
                /* high-word tap INPUT_BPF_TAPS - 2; MAC, prefetch w5 & w4 */
                "MAC w5*w4, A, [w9], w5, [w11], w4\n"
                /* high-word tap INPUT_BPF_TAPS - 1; MAC */
                "MAC w5*w4, A\n"
                /* disable modulo addressing */
                "CLR MODCON\n"
                /* return low-word */
                "MOV ACCAL, %0\n"
                /* return high-word */
                "MOV ACCAH, %d0"
                : "=r"(result)
                : "r"(tapsBy2Less2), "r"(coef), "r"(dly_line), "r"(dly_line_end_byte), "r"(en_modulo), "r"(newest_sample)
                : "w4", "w5", "w9", "w11");

    return result;

}

inline bool
lpf_decimate(uint16_t T_by_1024, _Q31_t stg3_coef, Q31_IQ_array_t input,
        __eds__ Q31_IQ_array_t *results)
{
    #include "AAF/AAF_STG12.h"

    static _Q31_t fdiff_I_stg1[AAF_STG1_DLY_SZ] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t fdiff_Q_stg1[AAF_STG1_DLY_SZ] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t fdiff_I_stg2[AAF_STG12_TAPS] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t fdiff_Q_stg2[AAF_STG12_TAPS] __attribute__((space(ymemory), eds, aligned));

    static _Q31_t f1_I_stg1[AAF_STG1_DLY_SZ] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t f1_Q_stg1[AAF_STG1_DLY_SZ] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t f1_I_stg2[AAF_STG12_TAPS] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t f1_Q_stg2[AAF_STG12_TAPS] __attribute__((space(ymemory), eds, aligned));

    static _Q31_t fsum_I_stg1[AAF_STG1_DLY_SZ] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t fsum_Q_stg1[AAF_STG1_DLY_SZ] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t fsum_I_stg2[AAF_STG12_TAPS] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t fsum_Q_stg2[AAF_STG12_TAPS] __attribute__((space(ymemory), eds, aligned));

#ifdef MEASURE_F2_AT_BRIDGE
    static _Q31_t f2_bridge_I_stg1[AAF_STG1_DLY_SZ] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t f2_bridge_Q_stg1[AAF_STG1_DLY_SZ] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t f2_bridge_I_stg2[AAF_STG12_TAPS] __attribute__((space(ymemory), eds, aligned));
    static _Q31_t f2_bridge_Q_stg2[AAF_STG12_TAPS] __attribute__((space(ymemory), eds, aligned));
#endif
```

```
#ifdef MEASURE_F2_AT_COIL
   static _Q31_t f2_coil_I_stg1[AAF_STG1_DLY_SZ] __attribute__((space(ymemory), eds, aligned));
   static _Q31_t f2_coil_Q_stg1[AAF_STG1_DLY_SZ] __attribute__((space(ymemory), eds, aligned));
   static _Q31_t f2_coil_I_stg2[AAF_STG12_TAPS] __attribute__((space(ymemory), eds, aligned));
   static _Q31_t f2_coil_Q_stg2[AAF_STG12_TAPS] __attribute__((space(ymemory), eds, aligned));
#endif

   static Q31_IQ_array_t stg3_tmp, result_static;
   static _Q31_t stg3_coef_loc;

   static uint8_t slice_stg1, slice_stg2, idx1, aligned_1, idx2, idx3, T_capture, slice_stg2_fsum, idx1_fsum, aligned_1_fsum,
idx2_fsum, idx3_fsum;
   static bool stg1_primed, stg1_started, stg2_primed, stg1_primed_fsum, stg1_started_fsum, stg2_primed_fsum;

   _Q31_t temp;

   if (0 == T_by_1024) {
      slice_stg1 = 0;
      slice_stg2 = 0;
      idx1 = 0;
      aligned_1 = 0;
      idx2 = 0;
      idx3 = 0;

      T_capture = 0;

      slice_stg2_fsum = 0;
      idx1_fsum = 0;
      aligned_1_fsum = 0;
      idx2_fsum = 0;
      idx3_fsum = 0;

      stg1_primed = 0;
      stg1_started = 0;
      stg2_primed = 0;

      stg1_primed_fsum = 0;
      stg1_started_fsum = 0;
      stg2_primed_fsum = 0;

      stg3_tmp.fdiff_I.Q31 = 0;
      stg3_tmp.fdiff_Q.Q31 = 0;
      stg3_tmp.f1_I.Q31 = 0;
      stg3_tmp.f1_Q.Q31 = 0;
      stg3_tmp.fsum_I.Q31 = 0;
      stg3_tmp.fsum_Q.Q31 = 0;
#ifdef MEASURE_F2_AT_BRIDGE
      stg3_tmp.bridge_f2_I.Q31 = 0;
      stg3_tmp.bridge_f2_Q.Q31 = 0;
#endif
#ifdef MEASURE_F2_AT_COIL
      stg3_tmp.coil_f2_I.Q31 = 0;
      stg3_tmp.coil_f2_Q.Q31 = 0;
#endif

      result_static.fdiff_I.Q31 = 0;
      result_static.fdiff_Q.Q31 = 0;
      result_static.f1_I.Q31 = 0;
      result_static.f1_Q.Q31 = 0;
      result_static.fsum_I.Q31 = 0;
      result_static.fsum_Q.Q31 = 0;
#ifdef MEASURE_F2_AT_BRIDGE
      result_static.bridge_f2_I.Q31 = 0;
      result_static.bridge_f2_Q.Q31 = 0;
#endif
#ifdef MEASURE_F2_AT_COIL
      result_static.coil_f2_I.Q31 = 0;
      result_static.coil_f2_Q.Q31 = 0;
#endif

      return false;
```

```
    }

    if (0 == T_capture) {
        T_capture = T_by_1024 & MASK_T;
        /*
         * low words of coefficients destined for stg3_mac need to be right-
         * shifted by 1 to avoid uint16_t x int16_t overflow; stg3_mac right-
         * shifts the low bits by 15, and thus the low word is correctly aligned
         */
        stg3_coef_loc.Q31 = stg3_coef.Q31;
    }

    /*
     * low words of samples destined for aaf_mpy need to be right shifted by 1
     * to avoid uint16_t x int16_t overflow; aaf_mpy right shifts the low bits
     * by 15, and thus the low word is correctly aligned
     */
    fdiff_I_stg1[idx1].dword.highWord = input.fdiff_I.dword.highWord;
    fdiff_I_stg1[idx1].dword.lowWord = input.fdiff_I.dword.lowWord >> 1;
    fdiff_Q_stg1[idx1].dword.highWord = input.fdiff_Q.dword.highWord;
    fdiff_Q_stg1[idx1].dword.lowWord = input.fdiff_Q.dword.lowWord >>1;

    fsum_I_stg1[idx1_fsum].dword.highWord = input.fsum_I.dword.highWord;
    fsum_I_stg1[idx1_fsum].dword.lowWord = input.fsum_I.dword.lowWord >> 1;
    fsum_Q_stg1[idx1_fsum].dword.highWord = input.fsum_Q.dword.highWord;
    fsum_Q_stg1[idx1_fsum].dword.lowWord = input.fsum_Q.dword.lowWord >> 1;

    f1_I_stg1[idx1].dword.highWord = input.f1_I.dword.highWord;
    f1_I_stg1[idx1].dword.lowWord = input.f1_I.dword.lowWord >> 1;
    f1_Q_stg1[idx1].dword.highWord = input.f1_Q.dword.highWord;
    f1_Q_stg1[idx1].dword.lowWord = input.f1_Q.dword.lowWord >> 1;


#ifdef MEASURE_F2_AT_BRIDGE
    f2_bridge_I_stg1[idx1].dword.highWord = input.bridge_f2_I.dword.highWord;
    f2_bridge_I_stg1[idx1].dword.lowWord = input.bridge_f2_I.dword.lowWord >> 1;
    f2_bridge_Q_stg1[idx1].dword.highWord = input.bridge_f2_Q.dword.highWord;
    f2_bridge_Q_stg1[idx1].dword.lowWord = input.bridge_f2_Q.dword.lowWord >> 1;
#endif

#ifdef MEASURE_F2_AT_COIL
    f2_coil_I_stg1[idx1].dword.highWord = input.coil_f2_I.dword.highWord;
    f2_coil_I_stg1[idx1].dword.lowWord = input.coil_f2_I.dword.lowWord >> 1;
    f2_coil_Q_stg1[idx1].dword.highWord = input.coil_f2_Q.dword.highWord;
    f2_coil_Q_stg1[idx1].dword.lowWord = input.coil_f2_Q.dword.lowWord >> 1;
#endif


    if (false == stg1_primed) {
        if (idx1 == AAF_STG12_TAPS_LESS1) {
            stg1_primed = true;
            aligned_1 = idx1;
        }
    }
    if (false == stg1_primed_fsum) {
        if (idx1_fsum == AAF_STG12_TAPS_LESS1) {
            stg1_primed_fsum = true;
            aligned_1_fsum = idx1_fsum;
        }
    }

    switch (slice_stg1) {
        case 0:
            if (true == stg1_primed) {
                if (false == stg1_started) {
                    stg1_started = true;
                }
                temp = aaf_mpy(fdiff_I_stg1,
                               (uint8_t*)fdiff_I_stg1 + AAF_STG1_DLY_END,
                               &fdiff_I_stg1[aligned_1], AAF_STG12,
                               AAF_STG12_TAPS_BY2_LESS2);
```

```
      /* need to right-shfit low word by 1 for aaf_mpy */
      temp.dword.lowWord = temp.dword.lowWord >> 1;
      fdiff_I_stg2[idx2] = temp;
   }
   break;

case 1:
   if (stg1_started) {
      temp = aaf_mpy(fdiff_Q_stg1,
                  (uint8_t *)fdiff_Q_stg1 + AAF_STG1_DLY_END,
                  &fdiff_Q_stg1[aligned_1], AAF_STG12,
                  AAF_STG12_TAPS_BY2_LESS2);
      /* need to right-shfit low word by 1 for aaf_mpy */
      temp.dword.lowWord = temp.dword.lowWord >> 1;
      fdiff_Q_stg2[idx2] = temp;
   }
   break;

case 2:
   if (stg1_started) {
      temp = aaf_mpy(f1_I_stg1,
                  (uint8_t *)f1_I_stg1 + AAF_STG1_DLY_END,
                  &f1_I_stg1[aligned_1], AAF_STG12,
                  AAF_STG12_TAPS_BY2_LESS2);
      /* need to right-shfit low word by 1 for aaf_mpy */
      temp.dword.lowWord = temp.dword.lowWord >> 1;
      f1_I_stg2[idx2] = temp;
   }
   break;

case 3:
   if (stg1_started) {
      temp = aaf_mpy(f1_Q_stg1,
                  (uint8_t *)f1_Q_stg1 + AAF_STG1_DLY_END,
                  &f1_Q_stg1[aligned_1], AAF_STG12,
                  AAF_STG12_TAPS_BY2_LESS2);
      /* need to right-shfit low word by 1 for aaf_mpy */
      temp.dword.lowWord = temp.dword.lowWord >> 1;
      f1_Q_stg2[idx2] = temp;
   }
   break;

case 4:
   if (stg1_primed_fsum) {
      if (false == stg1_started_fsum) {
         stg1_started_fsum = true;
      }
      temp = aaf_mpy(fsum_I_stg1,
                  (uint8_t *)fsum_I_stg1 + AAF_STG1_DLY_END,
                  &fsum_I_stg1[aligned_1_fsum], AAF_STG12,
                  AAF_STG12_TAPS_BY2_LESS2);
      /* need to right-shfit low word by 1 for aaf_mpy */
      temp.dword.lowWord = temp.dword.lowWord >> 1;
      fsum_I_stg2[idx2_fsum] = temp;
   }
   break;

case 5:
   if (stg1_started_fsum) {
      temp = aaf_mpy(fsum_Q_stg1,
                  (uint8_t *)fsum_Q_stg1 + AAF_STG1_DLY_END,
                  &fsum_Q_stg1[aligned_1_fsum], AAF_STG12,
                  AAF_STG12_TAPS_BY2_LESS2);
      /* need to right-shfit low word by 1 for aaf_mpy */
      temp.dword.lowWord = temp.dword.lowWord >> 1;
      fsum_Q_stg2[idx2_fsum] = temp;
   }
   break;

case 6:
   if (stg1_started) {
```

```
      aligned_1 += STG12_DECIMATION_FACTOR;
      if (aligned_1 >= AAF_STG1_DLY_SZ) {
        aligned_1 -= AAF_STG1_DLY_SZ;
      }
      if (false == stg2_primed) {
        if (AAF_STG12_TAPS_LESS1 == idx2) {
          stg2_primed = true;
        }
      }
    }
    if (stg1_started_fsum) {
      aligned_1_fsum += STG12_DECIMATION_FACTOR;
      if (aligned_1_fsum >= AAF_STG1_DLY_SZ) {
        aligned_1_fsum -= AAF_STG1_DLY_SZ;
      }
      if (false == stg2_primed_fsum) {
        if (AAF_STG12_TAPS_LESS1 == idx2_fsum) {
          stg2_primed_fsum = true;
        }
      }
    }
    break;

case 7:
  if (true == stg2_primed && idx3 < T_capture) {
      switch (slice_stg2) {
        case 0:
          stg3_tmp.fdiff_I = aaf_mpy(fdiff_I_stg2,
                    (uint8_t *)fdiff_I_stg2 + AAF_STG2_DLY_END,
                    &fdiff_I_stg2[idx2], AAF_STG12,
                    AAF_STG12_TAPS_BY2_LESS2);
          break;

        case 1:
          stg3_tmp.fdiff_Q = aaf_mpy(fdiff_Q_stg2,
                    (uint8_t *)fdiff_Q_stg2 + AAF_STG2_DLY_END,
                    &fdiff_Q_stg2[idx2], AAF_STG12,
                    AAF_STG12_TAPS_BY2_LESS2);
        break;

        case 2:
          stg3_tmp.f1_I = aaf_mpy(f1_I_stg2,
                   (uint8_t *)f1_I_stg2 + AAF_STG2_DLY_END,
                   &f1_I_stg2[idx2], AAF_STG12,
                   AAF_STG12_TAPS_BY2_LESS2);
          break;

        case 3:
          stg3_tmp.f1_Q = aaf_mpy(f1_Q_stg2,
                   (uint8_t *)f1_Q_stg2 + AAF_STG2_DLY_END,
                    &f1_Q_stg2[idx2], AAF_STG12,
                    AAF_STG12_TAPS_BY2_LESS2);
          break;

        case 4:
          result_static.fdiff_I = stg3_mac(result_static.fdiff_I,
                                   stg3_coef_loc,
                                   stg3_tmp.fdiff_I);
          result_static.fdiff_Q = stg3_mac(result_static.fdiff_Q,
                                   stg3_coef_loc,
                                   stg3_tmp.fdiff_Q);
          result_static.f1_I = stg3_mac(result_static.f1_I,
                                   stg3_coef_loc,
                                   stg3_tmp.f1_I);
          result_static.f1_Q = stg3_mac(result_static.f1_Q,
                                   stg3_coef_loc,
                                   stg3_tmp.f1_Q);

          ++idx3;
          break;
```

```
            default:
               break;
            }
            slice_stg2 = (slice_stg2 + 1) & 0x1F;
         }
         break;

      case 8:
         if (stg2_primed_fsum) {
            switch (slice_stg2_fsum) {
               case 0:
                  stg3_tmp.fsum_I = aaf_mpy(fsum_I_stg2,
                              (uint8_t *)fsum_I_stg2 + AAF_STG2_DLY_END,
                              &fsum_I_stg2[idx2_fsum], AAF_STG12,
                              AAF_STG12_TAPS_BY2_LESS2);
                  break;

               case 1:
                  stg3_tmp.fsum_Q = aaf_mpy(fsum_Q_stg2,
                              (uint8_t *)fsum_Q_stg2 + AAF_STG2_DLY_END,
                              &fsum_Q_stg2[idx2_fsum], AAF_STG12,
                              AAF_STG12_TAPS_BY2_LESS2);
                  break;

               case 2:
                  result_static.fsum_I = stg3_mac(result_static.fsum_I,
                                          stg3_coef_loc,
                                          stg3_tmp.fsum_I);
                  result_static.fsum_Q = stg3_mac(result_static.fsum_Q,
                                          stg3_coef_loc,
                                          stg3_tmp.fsum_Q);

                  ++idx3_fsum;
                  if (T_capture == idx3_fsum) {
                     *results = result_static;
                     return true;
                  }
                  break;

               default:
                  break;
            }
            slice_stg2_fsum = (slice_stg2_fsum + 1) & 0x1F;
         }
         break;

      case 9:
         if (stg1_started) {
            ++idx2;
            if (AAF_STG12_TAPS == idx2) {
               idx2 = 0;
            }
         }
         if (stg1_started_fsum) {
            ++idx2_fsum;
            if (AAF_STG12_TAPS == idx2_fsum) {
               idx2_fsum = 0;
            }
         }
         break;

      default:
         break;
   }

   ++idx1;
   if (AAF_STG1_DLY_SZ == idx1) {
      idx1 = 0;
   }

   if ((T_by_1024 & MAST_START_FSUM) != 0) {
```

```
        ++idx1_fsum;
        if (AAF_STG1_DLY_SZ == idx1_fsum) {
            idx1_fsum = 0;
        }
    }

    slice_stg1 = (slice_stg1 + 1) & 0x1F;

    return false;

}


/*
 * generateAndProcessSamples.c
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"
#include "spintronicsConfig.h"
#include "fsmStates.h"
#include "uartDrv.h"
#include "muxControl.h"
#include "balanceBridge.h"
#include "calculateVectors.h"
#include "commsDefines.h"
#include "utility.h"
#include "spintronicsStructs.h"
#include "FIR.h"
#include "constants.h"

_Q15 readBridgeSampleAndApplyGain(bool* bridgeDigitalClip);
static void signalGenerator(unsigned char runOrReset, angle_array_t *freqT, __eds__ _Q15 *cosOmega1T, __eds__ _Q15
*cosOmega2T, _Q15 local_a2, _Q15 local_f2);
bool measure(uint16_t T_by_1024, _Q31_t stg3_coef, _Q15 bridgeSample,
#ifdef MEASURE_F2_AT_COIL
            _Q15 coilSample,
#endif
             angle_array_t *freqT, _Q15 cosOmega1T,
#if defined(MEASURE_F2_AT_BRIGE) || defined(MEASURE_F2_AT_COIL)
             _Q15 cosOmega2T,
#endif
             __eds__ Q31_IQ_array_t *results);

extern _Q15 _Q15cosPILUT(_Q15 phiOverPI);
extern _Q15 _Q15sinPILUT(_Q15 phiOverPI);
extern _Q31_t _Q23cosPILUT(_Q15 phiOverPI);
extern _Q31_t Q23xQ15Mult(register _Q31_t input, register _Q15 amplitude);

void measurementFSM(void)
{
    volatile _Q15 bridgeSample;
    volatile _Q15 coilSample;

    bool balanceBridgeFirst = false;

    static uint8_t sensorAddress;
    static uint32_t timer;
    static uint8_t sensorIndex;
    _Q15 cosOmega1T;//fractions of full-scale
    _Q15 cosOmega2T;//fractions of full-scale
    static angle_array_t freqT;//array contents: 2*f1*t, 2*f2*t, 2*fdiff*t, 2*fsum*t
    Q31_IQ_array_t results;
    static bool bridgeADCClip;
```

```
    static bool coilADCClip;
    static bool bridgeDigitalClip;
    static _Q15 current_a2 = 0;

    //local copies of globals
    static _Q15 local_f2;//units are Q15 half-cycles per sample-period
    static _Q15 local_a2;


#ifdef SIMULATION_MODE
    uint16_t RXBUF2 = rand();//only to give random stimulus during simulation
#endif


    if (RXBUF0 == 0x7FFF || RXBUF0 == 0x8000)
    {
        bridgeADCClip = true;
    }
    if (RXBUF2 == 0x7FFF || RXBUF2 == 0x8000)
    {
        coilADCClip = true;
    }
    coilSample = RXBUF2;

    switch (global_state)
    {

        case RAMP_DOWN_COIL_QUIT:

            if (current_a2 != 0) {
                --current_a2;
            }
            signalGenerator(RESET_BRIDGE_GEN, &freqT, &cosOmega1T, &cosOmega2T, current_a2, local_f2);
            if (0 == current_a2) {
                global_state = IDLE;
            }
            break;


        case RAMP_DOWN_COIL_RESTART:

            if (current_a2 != 0) {
                --current_a2;
            }
            signalGenerator(RESET_BRIDGE_GEN, &freqT, &cosOmega1T, &cosOmega2T, current_a2, local_f2);
            if (0 == current_a2) {
                global_state = START_MEASUREMENT_FSM;
            }
            break;


        case RAMP_DOWN_COIL_BALANCE_BRIDGE:

            if (current_a2 != 0) {
                --current_a2;
            }
            signalGenerator(RESET_BRIDGE_GEN, &freqT, &cosOmega1T, &cosOmega2T, current_a2, local_f2);
            if (0 == current_a2) {
                global_state = START_BRIDGE_BALANCE_FSM;
            }
            break;


        case START_MEASUREMENT_FSM:

            if (!sensorRBridgeTableValid) {
                balanceBridgeFirst = false;//true;//TEMPORARY, to bypass bridge balance
            }

            /*
             * capture f2/a2 so that coil signal can ramp down when a new START command
```

```
         * interrupts the current measurment
         */
        local_f2 = f2;
        local_a2 = a2;


        if (balanceBridgeFirst) {

           global_state = START_BRIDGE_BALANCE_FSM | START_MEASUREMENT_AFTER_BALANCE_MASK;

        } else {
           sensorIndex = 0;
           signalGenerator(RESET_SIGNAL_GEN, &freqT, &cosOmega1T, &cosOmega2T, current_a2, local_f2);
           global_state = RAMP_UP_COIL;
           current_a2 = 0;
           setRAmp(u24_code);
        }
        break;


      case RAMP_UP_COIL:

        signalGenerator(RESET_BRIDGE_GEN, &freqT, &cosOmega1T, &cosOmega2T, current_a2, local_f2);
        ++current_a2;
        if (current_a2 == local_a2) {
           global_state = START_NEW_MEASUREMENT_CYCLE;
        }
        break;


      case START_NEW_MEASUREMENT_CYCLE:
      {
        uint16_t r_bridge;

        timer = 0;
        signalGenerator(RESET_BRIDGE_GEN, &freqT, &cosOmega1T, &cosOmega2T, current_a2, local_f2);

        /* reset for a new measurement */
        measure(0, stg3_coef, 0,
#ifdef MEASURE_F2_AT_COIL
                0,
#endif
                &freqT, cosOmega1T,
#if defined(MEASURE_F2_AT_BRIGE) || defined(MEASURE_F2_AT_COIL)
                cosOmega2T,
#endif
                &results);

        bridgeADCClip = false;
        coilADCClip = false;
        bridgeDigitalClip = false;

        sensorAddress = sensorAddressTable[sensorIndex];
        r_bridge = sensorRBridgeTable[sensorIndex];

        configSensor(sensorAddress);
        setRBridge(r_bridge);

#ifndef SIMULATION_MODE
        global_state = WAIT_FOR_COIL_0RAD;
#else
        global_state = START_SIGNAL_GEN;
#endif
        break;
      }


      case WAIT_FOR_COIL_0RAD:

        /* start reading samples before MEASURE for FIR group delay */
        bridgeSample = readBridgeSampleAndApplyGain(&bridgeDigitalClip);
```

```
        signalGenerator(RESET_BRIDGE_GEN, &freqT, &cosOmega1T, &cosOmega2T, current_a2, local_f2);
        if (0 == freqT.two_f2_t)
        {
          timer = 0;
          global_state = START_SIGNAL_GEN;
        }
        break;


      case START_SIGNAL_GEN:

        /* start reading samples before MEASURE for FIR group delay */
        bridgeSample = readBridgeSampleAndApplyGain(&bridgeDigitalClip);
        signalGenerator(RUN_SIGNAL_GEN, &freqT, &cosOmega1T, &cosOmega2T, current_a2, local_f2);
        ++timer;
        if (timer == MEASUREMENT_SETUP_TIME)
        {
          timer = 0;
          if (0 == DELAY_TO_60HZ_PI) {
            global_state = MEASURE;
          } else {
            global_state = MEASURE_NOT_FSUM;
          }
        }
        break;


      case MEASURE_NOT_FSUM:
      {
        bridgeSample = readBridgeSampleAndApplyGain(&bridgeDigitalClip);

        measure(measurementTime | MASK_START_NOT_FSUM, stg3_coef,
            bridgeSample,
#ifdef MEASURE_F2_AT_COIL
            coilSample,
#endif
            &freqT, cosOmega1T,
#if defined(MEASURE_F2_AT_BRIGE) || defined(MEASURE_F2_AT_COIL)
            cosOmega2T,
#endif
            &results);
        signalGenerator(RUN_SIGNAL_GEN, &freqT, &cosOmega1T, &cosOmega2T, current_a2, local_f2);

        ++timer;
        if (DELAY_TO_60HZ_PI == timer)
        {
          timer = 0;
          global_state = MEASURE;
        }
        break;
      }


      case MEASURE:
      {
        bool measurement_finished;
        bridgeSample = readBridgeSampleAndApplyGain(&bridgeDigitalClip);

        measurement_finished = measure(measurementTime | MASK_START_EVERYTHING,
            stg3_coef,
            bridgeSample,
#ifdef MEASURE_F2_AT_COIL
            coilSample,
#endif
            &freqT, cosOmega1T,
#if defined(MEASURE_F2_AT_BRIGE) || defined(MEASURE_F2_AT_COIL)
            cosOmega2T,
#endif
            &results);
        signalGenerator(RUN_SIGNAL_GEN, &freqT, &cosOmega1T, &cosOmega2T, current_a2, local_f2);
```

```
            if (true == measurement_finished)
            {
               global_state = CALCULATE_VECTORS;
            }
            break;
         }

         case CALCULATE_VECTORS:

            signalGenerator(RESET_BRIDGE_GEN, &freqT, &cosOmega1T, &cosOmega2T, current_a2, local_f2);

            //capture necessary variables for processing the vector calc thread
            sensorAddressCapture = sensorAddress;
            IQ_capture = results;
            bridgeADCClipCapture = bridgeADCClip;
            coilADCClipCapture = coilADCClip;
            bridgeDigitalClipCapture = bridgeDigitalClip;
            f1PlusF2OutOfRangeCapture = f1PlusF2OutOfRange;
            inverseBridgeAnalogGainCapture = inverseBridgeAnalogGain;

            //start the timer to spawn the vector calc thread
            PR1 = DELAY_TO_VECTOR_CALC_THREAD;
            T1CONbits.TON = 1;

#ifdef SIMULATION_MODE
            calculateFinalVectors();
#endif

            ++sensorIndex;
            if (sensorIndex >= numberOfSensors) {

               /*
                * must test >= in case numberOfSensors was updated since
                * the last iteration of this FSM
                */
               sensorIndex = 0;
            }
            global_state = START_NEW_MEASUREMENT_CYCLE;
            break;


         default:
            global_state = IDLE;
            break;
      }

}

_Q15 readBridgeSampleAndApplyGain(bool* bridgeDigitalClip)
{

#ifdef SIMULATION_MODE
      uint16_t RXBUF0 = rand();//only to give random stimulus during simulation
      uint16_t RXBUF1 = rand();//only to give random stimulus during simulation
#endif

      if (bridgeADCGainFactor == 0)
      {
         return FIR(RXBUF0);
      }
      else
      {
         int16_t clipTest;
         int16_t maskForTruncationBitsPlusOne = 0x8000;//make it signed so that the shift uses sign extension
         maskForTruncationBitsPlusOne = maskForTruncationBitsPlusOne >> bridgeADCGainFactor;
         clipTest = RXBUF0 & maskForTruncationBitsPlusOne;
         if (clipTest != 0x0000 && clipTest != maskForTruncationBitsPlusOne)
         {
            *bridgeDigitalClip = true;
            if ((int16_t)RXBUF0 < 0)
            {
```

```
            return FIR(0x8000);//interpret the clipped sample as the most negative value
          }
          else
          {
            return FIR(0x7FFF);//interpret the clipped sample as the most positive value
          }
        }
        else
        {
          _Q31_t tempSample;
          tempSample.dword.lowWord = RXBUF1;
          tempSample.dword.highWord = RXBUF0;
          tempSample.Q31 = tempSample.Q31 << bridgeADCGainFactor;
          return FIR(tempSample.Q23_aligned.Q15);
        }
    }
}

static inline void
signalGenerator(uint8_t runOrReset, angle_array_t *freqT,
              __eds__ _Q15 *cosOmega1T, __eds__ _Q15 *cosOmega2T,
              _Q15 current_a2, _Q15 local_f2)
{

    _Q31_t cosOmega1T_24bit;
    _Q31_t cosOmega2T_24bit;
    if (runOrReset == RUN_SIGNAL_GEN) {

        cosOmega1T_24bit = _Q23cosPILUT(freqT->two_f1_t);//generating cos(omega1 * t)
        cosOmega2T_24bit = _Q23cosPILUT(freqT->two_f2_t);//generating cos(omega2 * t)

        *cosOmega1T = cosOmega1T_24bit.Q23_aligned.Q15;
        *cosOmega2T = cosOmega2T_24bit.Q23_aligned.Q15;

        if (a1 < 0x7FFF) {
            cosOmega1T_24bit = Q23xQ15Mult(cosOmega1T_24bit, a1);
        }

        TXBUF0 = cosOmega1T_24bit.Q23_aligned.Q15;
        TXBUF1 = cosOmega1T_24bit.dword.lowWord;

        #ifndef PROBE
        if (current_a2 < 0x7FFF) {
            cosOmega2T_24bit = Q23xQ15Mult(cosOmega2T_24bit, current_a2);
        }

        TXBUF2 = cosOmega2T_24bit.Q23_aligned.Q15;
        TXBUF3 = cosOmega2T_24bit.dword.lowWord;

        #elif defined(PROBE_BRIDGE_ADC)
        TXBUF2 = RXBUF0;
        TXBUF3 = RXBUF1;
        #elif defined(PROBE_COIL_ADC)
        TXBUF2 = RXBUF2;
        TXBUF3 = RXBUF3;
        #endif

        freqT->two_f1_t += f1;
        freqT->two_f2_t += local_f2;
        freqT->two_fdiff_t += fdiff;
        freqT->two_fsum_t += fsum;

    } else if (runOrReset == RESET_BRIDGE_GEN) {

        cosOmega2T_24bit = _Q23cosPILUT(freqT->two_f2_t);//generating cos(omega2 * t)
        *cosOmega2T = cosOmega2T_24bit.Q23_aligned.Q15;

        #ifndef PROBE
        if (current_a2 < 0x7FFF) {
            cosOmega2T_24bit = Q23xQ15Mult(cosOmega2T_24bit, current_a2);
        }
```

```
        TXBUF2 = cosOmega2T_24bit.Q23_aligned.Q15;
        TXBUF3 = cosOmega2T_24bit.dword.lowWord;

        #elif defined(PROBE_BRIDGE_ADC)
        TXBUF2 = RXBUF0;
        TXBUF3 = RXBUF1;
        #elif defined(PROBE_COIL_ADC)
        TXBUF2 = RXBUF2;
        TXBUF3 = RXBUF3;
        #endif

        freqT->two_f2_t += local_f2;

        *cosOmega1T = 0;
        TXBUF0 = 0x0000;
        TXBUF1 = 0x0000;

        freqT->two_f1_t = 0x0000;
        freqT->two_fdiff_t = 0x0000;
        freqT->two_fsum_t = 0x0000;

    } else {

        *cosOmega1T = 0;
        *cosOmega2T = 0;
        TXBUF0 = 0x0000;
        TXBUF1 = 0x0000;
        TXBUF2 = 0x0000;
        TXBUF3 = 0x0000;

        freqT->two_f1_t = 0x0000;
        freqT->two_f2_t = 0x0000;
        freqT->two_fdiff_t = 0x0000;
        freqT->two_fsum_t = 0x0000;
    }
}

bool measure(uint16_t T_by_1024, _Q31_t stg3_coef, _Q15 bridgeSample,
#ifdef MEASURE_F2_AT_COIL
            _Q15 coilSample,
#endif
             angle_array_t *freqT, _Q15 cosOmega1T,
#if defined(MEASURE_F2_AT_BRIGE) || defined(MEASURE_F2_AT_COIL)
             _Q15 cosOmega2T,
#endif
             __eds__ Q31_IQ_array_t *results)
{
    Q31_IQ_array_t input;

    _Q15 cosFDiffT = _Q15cosPILUT(freqT->two_fdiff_t);
    _Q15 cosFSumT = _Q15cosPILUT(freqT->two_fsum_t);
    _Q15 sinF1T = _Q15sinPILUT(freqT->two_f1_t);
#if defined(MEASURE_F2_AT_BRIGE) || defined(MEASURE_F2_AT_COIL)
    _Q15 sinF2T = _Q15sinPILUT(freqT->two_f2_t);
#endif
    _Q15 sinFDiffT = _Q15sinPILUT(freqT->two_fdiff_t);
    _Q15 sinFSumT = _Q15sinPILUT(freqT->two_fsum_t);

    input.f1_I.Q31 = asm16X16Mult(bridgeSample, cosOmega1T);
#ifdef MEASURE_F2_AT_BRIDGE
    input.bridge_f2_I.Q31 = asm16X16Mult(bridgeSample, cosOmega2T);
#endif
    input.fdiff_I.Q31 = asm16X16Mult(bridgeSample, cosFDiffT);
    input.fsum_I.Q31 =  asm16X16Mult(bridgeSample, cosFSumT);
#ifdef MEASURE_F2_AT_COIL
    input.coil_f2_I.Q31 = asm16X16Mult(coilSample, cosOmega2T);
#endif
    input.f1_Q.Q31 = asm16X16Mult(bridgeSample, sinF1T);
#ifdef MEASURE_F2_AT_BRIDGE
    input.bridge_f2_Q.Q31 = asm16X16Mult(bridgeSample, sinF2T);
```

```
#endif
   input.fdiff_Q.Q31 = asm16X16Mult(bridgeSample, sinFDiffT);
   input.fsum_Q.Q31 = asm16X16Mult(bridgeSample, sinFSumT);
#ifdef MEASURE_F2_AT_COIL
   input.coil_f2_Q.Q31 = asm16X16Mult(coilSample, sinF2T);
#endif

   return lpf_decimate(T_by_1024, stg3_coef, input, results);

#if defined(PROBE) && defined(PROBE_BRIDGE_X_COS_F1_T)
   TXBUF2 = input.f1_I.dword.highWord;
   TXBUF3 = input.f1_I.dword.lowWord;
#elif defined(PROBE) && defined(PROBE_BRIDGE_X_COS_F2_T) && defined(MEASURE_F2_AT_BRIDGE)
   TXBUF2 = input.bridge_f2_I.dword.highWord;
   TXBUF3 = input.bridge_f2_I.dword.lowWord;
#elif defined(PROBE) && defined(PROBE_BRIDGE_X_COS_FDIFF_T)
   TXBUF2 = input.fdiff_I.dword.highWord;
   TXBUF3 = input.fdiff_I.dword.lowWord;
#elif defined(PROBE) && defined(PROBE_BRIDGE_X_COS_FSUM_T)
   TXBUF2 = input.fsum_I.dword.highWord;
   TXBUF3 = input.fsum_I.dword.lowWord;
#elif defined(PROBE) && defined(PROBE_COIL_X_COS_F2_T)
   TXBUF2 = input.coil_f2_I.dword.highWord;
   TXBUF3 = input.coil_f2_I.dword.lowWord;
#elif defined(PROBE) && defined(PROBE_BRIDGE_X_SIN_F1_T)
   TXBUF2 = input.f1_Q.dword.highWord;
   TXBUF3 = input.f1_Q.dword.lowWord;
#elif defined(PROBE) && defined(PROBE_BRIDGE_X_SIN_F2_T) && defined(MEASURE_F2_AT_BRIDGE)
   TXBUF2 = input.bridge_f2_Q.dword.highWord;
   TXBUF3 = input.bridge_f2_Q.dword.lowWord;
#elif defined(PROBE) && defined(PROBE_BRIDGE_X_SIN_FDIFF_T)
   TXBUF2 = input.fdiff_Q.dword.highWord;
   TXBUF3 = input.fdiff_Q.dword.lowWord;
#elif defined(PROBE) && defined(PROBE_BRIDGE_X_SIN_FSUM_T)
   TXBUF2 = input.fsum_Q.dword.highWord;
   TXBUF3 = input.fsum_Q.dword.lowWord;
#elif defined(PROBE) && defined(PROBE_COIL_X_SIN_F2_T)
   TXBUF2 = input.coil_f2_Q.dword.highWord;
   TXBUF3 = input.coil_f2_Q.dword.lowWord;
#endif
}


/*
 * i2sDrv.c
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"
#include "fsmStates.h"
#include "generateAndProcessSamples.h"
#include "balanceBridge.h"
#include "uartDrv.h"

/*
i2sInit(): Initialise DCI for I2S
*/

void i2sInit (void)
{

   //make RE5, RE6, RE7 digital pins
   ANSELEbits.ANSE5 = 0;
   ANSELEbits.ANSE6 = 0;
```

```
    ANSELEbits.ANSE7 = 0;

    //make RG6 a digital pin
    ANSELGbits.ANSG6 = 0;

    RPINR25bits.COFSR = 118;// route PIN4/RP118/RG6 to COFS
    RPINR24bits.CSCKR = 87; // route PIN3/RP87/RE7 to CSCK
    RPINR24bits.CSDIR = 86; // route PIN2/RP86/RE6 to CSDI
    RPOR6bits.RP85R = 0b001011;// route CSDO to PIN1/RP85/RE5

// DCI Control Register DCICON1 Initialization
    DCICON1bits.DCISIDL = 0;// module operates in CPU Idle Mode
    DCICON1bits.DLOOP = 0;  // Digital Loopback disabled
    DCICON1bits.CSCKD = 1;  // CSCK is an input
    DCICON1bits.CSCKE = 1;  // data sampled on sck rising edge
    DCICON1bits.COFSD = 1;  // COFS is an input
    DCICON1bits.UNFM = 0;   // xmit 0 on underflow
    DCICON1bits.CSDOM = 0;  // CSDO drives 0 during disabled time slots
#if defined(CODEC_USES_I2S)
    DCICON1bits.DJST = 0;   // xmit/rx begins on serial clock after fsync
#else
    DCICON1bits.DJST = 1;   // xmit/rx begins same serial clock as fsync
#endif
    DCICON1bits.COFSM = 1;  // i2s mode

// DCI Control Register DCICON2 Initialization
    DCICON2bits.BLEN = 3;   // 4 words = 64 bits buffered between ineterupts
    DCICON2bits.COFSG = 1;  // data frame has 4 words = 64 bits
    DCICON2bits.WS = 15;    // data word size is 16 bits

// DCI Control Register DCICON3 Initialization
    DCICON3 = 0;            //serial clock provided externally

// Transmit Slot Control Register Initialization
    TSCONbits.TSE0 = 1;     // Transmit on Time Slot 0
    TSCONbits.TSE1 = 1;             // Transmit on Time Slot 1

// Receiver Slot Control Register Initialization
    RSCONbits.RSE0 = 1;             // Receive on Time Slot 0
    RSCONbits.RSE1 = 1;             // Receive on Time Slot 1


// Initialize the TX buffers
    TXBUF0 = 0x0000;
    TXBUF1 = 0x0000;
    TXBUF2 = 0x0000;
    TXBUF3 = 0x0000;

// Enable DCI module
    IPC15bits.DCIIP = 6;    // Set the interrupt priority
    IFS3bits.DCIIF = 0;     // Clear the interrupt flag
    IEC3bits.DCIIE = 1;     // Enable the interrupt
    DCICON1bits.DCIEN = 1;  //DCI Module Enabled
}

void __attribute__((__interrupt__, no_auto_psv)) _DCIInterrupt(void)
{

    IFS3bits.DCIIF = 0;

    if (global_state & BALANCE_BRIDGE_FSM_MASK) {
        balanceBridgeFSM();
    } else {
        measurementFSM();
    }
}


/*
 * main.c
 *
```

```
 * designed and written
 * by the Spintronics Senior Design Embedded Team:
 *     Jonathan Pechuman
 *     Hajime Makino
 *     Samiha Sultana
 *     Michael Sandstedt
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"
#include "spintronicsConfig.h"
#include "timerDrv.h"
#include "spiTx.h"
#include "CS4272.h"
#include "muxControl.h"
#include "uartDrv.h"
#include "FIR.h"

#ifdef SIMULATION_MODE
#include "balanceBridge.h"
#include "fsmStates.h"
#endif
#ifdef NO_GUI
#include "uartDrv.h"
#endif

// Select Internal FRC at POR
_FOSCSEL(FNOSC_FRC & IESO_OFF);               // OSC2 Pin Function: OSC2 is Clock Output
// Primary Oscillator Mode: Disabled
// Enable Clock Switching and Configure POSC in XT mode
_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF & POSCMD_XT);
_FWDT(FWDTEN_OFF);                            // Watchdog Timer Enabled/disabled by user software
                                                                        // (LPRC can be disabled by clearing SWDTEN bit in RCON
register
_FPOR(FPWRT_PWR1 & ALTI2C1_ON );                                // Turn off the power-up timers.
//_FGS(GCP_OFF);                              // Disable Code Protection

int main(void)
{
   // Configure Oscillator to operate the device at 140Mhz
   CLKDIVbits.PLLPRE = 0;
   CLKDIVbits.PLLPOST = 0;
   PLLFBDbits.PLLDIV = 110;       //PLL 5MHz to 140 MHz

   // Initiate Clock Switch to Primary Oscillator with PLL (NOSC=0b011)
   __builtin_write_OSCCONH(0x03);
   __builtin_write_OSCCONL(OSCCON | 0x01);

   // Wait for Clock switch to occur
   while (OSCCONbits.COSC!= 0b011);
   // Wait for PLL to lock
   while (OSCCONbits.LOCK!= 1);

   // Disable Watch Dog Timer
   RCONbits.SWDTEN=0;

   //initialize components; order is important; DO NOT CHANGE
   firInit();
   timerInit();//initialize the timer
   spiInit();// init SPI for controlling digi pots
   cs4272Init();// establish communication with the CS4272
   muxInit();// setup pins to communicate with the multiplexer
   uart_Init();// Init UART for GUI communication

#if defined(NO_GUI) || defined(SIMULATION_MODE)
      numberOfSensors = 5;
      sensorAddressTable[0] = 0x2F;
```

```
        sensorAddressTable[1] = 47;
        sensorAddressTable[2] = 45;
        sensorAddressTable[3] = 16;
        sensorAddressTable[4] = 17;
        processStartCommand(/*a1*/.300, /*f1*/1000.0, /*a2*/.05, /*f2*/100.0,
                        /*T*/0.5, /*digital gain*/1, /*analog gain*/20.0);
#endif

#ifdef SIMULATION_MODE
    global_state = RAMP_DOWN_COIL_RESTART;
    sensorRBridgeTableValid = true;
#endif


    while(1)
    {
#ifdef SIMULATION_MODE
        measurementFSM();
#endif
    }
}


/*
 * muxControl.c
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 * and Todd Klein
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"

void muxInit (void)
{
    /*
     * mux connections:
     * A0  <- RD9
     * A1  <- RD10
     * A2  <- RD11
     * A3  <- RD0
     * EN0 <- RD1
     * EN1 <- RD2
     *
     * None are analog-capable, so no need to set ANSEL
     */

    //set relevant PORTD pins as outputs
    TRISDbits.TRISD0 = 0;
    TRISDbits.TRISD1 = 0;
    TRISDbits.TRISD2 = 0;
    TRISDbits.TRISD9 = 0;
    TRISDbits.TRISD10 = 0;
    TRISDbits.TRISD11 = 0;

    //make the relevant pins open drain / 5V tolerant
    ODCDbits.ODCD0 = 1;
    ODCDbits.ODCD1 = 1;
    ODCDbits.ODCD2 = 1;
    ODCDbits.ODCD9 = 1;
    ODCDbits.ODCD10 = 1;
    ODCDbits.ODCD11 = 1;

    //clear the pins
    PORTDbits.RD0 = 0;
    PORTDbits.RD1 = 0;
```

```
    PORTDbits.RD2 = 0;
    PORTDbits.RD9 = 0;
    PORTDbits.RD10 = 0;
    PORTDbits.RD11 = 0;
}

void configSensor(uint8_t sensor)
{
    if(sensor & 0x01) {
        PORTDbits.RD9 = 1;
    } else {
        PORTDbits.RD9 = 0;
    }

    if(sensor & 0x02) {
        PORTDbits.RD10 = 1;
    } else {
        PORTDbits.RD10 = 0;
    }

    if(sensor & 0x04) {
        PORTDbits.RD11 = 1;
    } else {
        PORTDbits.RD11 = 0;
    }

    if(sensor & 0x08) {
        PORTDbits.RD0 = 1;
    } else {
        PORTDbits.RD0 = 0;
    }

    if(sensor & 0x10) {
        PORTDbits.RD1 = 1;
    } else {
        PORTDbits.RD1 = 0;
    }

    if(sensor & 0x20) {
        PORTDbits.RD2 = 1;
    } else {
        PORTDbits.RD2 = 0;
    }

    //top two bits of sensor address are ignored (not connected in hardware)

}



/*
 * File:   Q23xQ15Mult.c
 * Author: Michael Sandstedt
 *
 * Created on December 19, 2013, 7:02 PM
 */

#include "spintronicsStructs.h"
#include "spintronicsIncludes.h"

/* multiply a _Q23 packed in a _Q31 by a _Q15 */
_Q31_t Q23xQ15Mult(register _Q31_t input, register _Q15 amplitude)
{

    volatile register _Q31_t result;

    __asm__ volatile ("MOV %d1, w4\n"
                      "MOV %2, w5\n"
                      "MPY w4*w5, A\n"
                      "LSR %1, #8, w4\n"
                      "ASR w5, #8, w5\n"
```

```
                "MAC w4*w5, A\n"
                "MOV ACCAH, %d0\n"
                "MOV ACCAL, %0"
                : "=r"(result)
                : "r"(input), "r"(amplitude)
                : "w4", "w5");

    return result;

}


/*
 * spiTx.c
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 *
 * First release: September 10th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"
#include "spiTx.h"
#include "utility.h"

#define CS1 PORTBbits.RB0
#define CS2 PORTBbits.RB1
#define CS3 PORTBbits.RB2
#define ALL_CS_MASK = 0x0007

#define SPI_TX_BUF_SIZE 32

/*
 * top 4 bits of each SPI_TX_BUF 16-bit word are the address,
 * this mask is just to mask off bytes though
 */
#define SPI_TX_ADDR_MASK 0xF0
/*
 * bottom 12 bits of each SPI_TX_BUF 16-bit word are the payload
 */
#define SPI_TX_PL_MASK 0x0FFF

//function prototypes
static void spiTxWorker(void);

//static variables
static uint8_t spiTxCur, spiTxEnd;
static uint16_t spiTxBuf [SPI_TX_BUF_SIZE];

void spiInit(void)
{
    uint16_t junk;

    spiTxCur = 0;
    spiTxEnd = 0;

    // RE2, RE4 to be digital pins
    ANSELEbits.ANSE2 = 0;
    ANSELEbits.ANSE4 = 0;

    // RB0, RB1, RB2 to be digital pins
    ANSELBbits.ANSB0 = 0;
    ANSELBbits.ANSB1 = 0;
    ANSELBbits.ANSB2 = 0;

    // Disable internal pullups for RB0, RB1, RB2
    CNPUBbits.CNPUB0 = 0;
    CNPUBbits.CNPUB1 = 0;
```

```
    CNPUBbits.CNPUB2 = 0;

    RPOR5bits.RP82R = 0b000101;// SDO1 (board designator SDI)
    RPOR5bits.RP84R = 0b000110;// SCK1 (board designator CLK)
    TRISBbits.TRISB0 = 0;// set as output (board designator CS1)
    TRISBbits.TRISB1 = 0;// set as output (board designator CS2)
    TRISBbits.TRISB2 = 0;// set as output (board designator CS3)

    //Disable all chip select lines
    CS1 = 1;// PORTBbits.RB0 = 1
    CS2 = 1;// PORTBbits.RB1 = 1
    CS3 = 1;// PORTBbits.RB2 = 1

    IFS0bits.SPI1IF = 0;// Clear the Interrupt flag
    IEC0bits.SPI1IE = 0;// Disable the interrupt
    SPI1STATbits.SISEL = 0b101;// Interrupt when the last bit is shifted out of SPIxSR
    IPC2bits.SPI1IP = 4;// set SPI1 interrupt priority to 4
    SPI1CON1bits.MSTEN = 1;// Master Mode
    SPI1CON1bits.DISSCK = 0;// Internal SPI1 clock is enabled
    SPI1CON1bits.DISSDO = 0;// SDO1 pin is controlled by the module
    SPI1CON1bits.MODE16 = 1;// Communication is word-wide (16 bits)//must shift at least 10 bits at a time to AD8400
    SPI1CON1bits.CKP = 0;// Idle state for clock is low level
    SPI1CON1bits.CKE = 1;// Serial output data changes on transition from active clock state to idle clock state
    SPI1CON1bits.SSEN = 0;// SS1 pin is not used by the module, pin is controlled by port function
    //set SCK to 70MHz / (1 * 16) = 4.375MHz
    SPI1CON1bits.SPRE = 0x7;    //Sec. prescale = 1:1
    SPI1CON1bits.PPRE = 0x1;    //Prim. prscale = 16:1
    SPI1CON2bits.FRMEN = 0;// Framed support is disabled
    SPI1STATbits.SPIROV = 0;// Clear the SPIROV bit
    SPI1CON2bits.SPIBEN = 0;// Enhanced Buffer is disabled
    SPI1STATbits.SPISIDL = 0;// Continue the module operation in Idle mode
    SPI1STATbits.SPIEN = 1;// enable SPI module
    IFS0bits.SPI1IF = 0;// Clear the Interrupt flag
    junk = SPI1BUF;// clear the buffer of all input
}

void __attribute__((__interrupt__, no_auto_psv)) _SPI1Interrupt(void)
{
    uint16_t junk;

    IFS0bits.SPI1IF = 0;// Clear the Interrupt flag
    IEC0bits.SPI1IE = 0;// Disable the interrupt to handle CS and more transmission

    //Disable all chip select lines
    CS1 = 1;
    CS2 = 1;
    CS3 = 1;

    SPI1STATbits.SPIROV = 0;// Clear the SPIROV bit

    //clear the buffer of all input
    junk = SPI1BUF;

    //AD5160 requires > 40ns for CS low-high setup before latching input
    NOP();
    NOP();
    NOP();

    START_ATOMIC();//begin critical section; must be atomic!
    ++spiTxCur;
    if (spiTxCur == SPI_TX_BUF_SIZE) {
        spiTxCur = 0;
    }
    if (spiTxCur != spiTxEnd) {
        spiTxWorker();
    }
    END_ATOMIC();//end critical section
}

static void spiTxWorker(void)
{
```

```
    uint8_t addr;

    START_ATOMIC();//begin critical section; must be atomic!
    addr = *((uint8_t *)(spiTxBuf + spiTxCur) + 1) & SPI_TX_ADDR_MASK;

    switch (addr) {
       case U20:
          CS1 = 0;
          //AD5160 requires > 15ns for CS high-low setup before sck starts
          NOP();
          NOP();
          break;
       case U23:
          CS2 = 0;
          //AD8400 requires > 10ns for CS high-low setup before sck starts
          NOP();
          break;
       case U24:
          CS3 = 0;
          //AD8400 requires > 10ns for CS high-low setup before sck starts
          NOP();
          break;
       default:
          break;
    }

    SPI1BUF = spiTxBuf[spiTxCur] & SPI_TX_PL_MASK;//only grab the lowest 12 bits
    END_ATOMIC();//end critical section

    IEC0bits.SPI1IE = 1;// Enable the interrupt to handle CS and more transmission

}

void spiTx(uint8_t addr, uint8_t pl)
{
    bool spawnTxThread = false;
    uint8_t bufSpaceAvl;

    START_ATOMIC();//begin critical section; must be atomic!

    if (spiTxCur == spiTxEnd) {
       spawnTxThread = true;
       bufSpaceAvl = SPI_TX_BUF_SIZE - 1;
    } else if (spiTxEnd > spiTxCur) {
       bufSpaceAvl = SPI_TX_BUF_SIZE - 1 - (spiTxEnd - spiTxCur);
    }
    else {
       bufSpaceAvl = spiTxCur - spiTxEnd - 1;
    }

    if (bufSpaceAvl > 0) {
       /*
        * We will use the top 4 bits of the payload as the address for CS;
        * this allows us to shift up to 12 bits per transfer.
        * The AD8400 digipot requires a 10 bit code, where the two MSB are 0.
        * Thus, more than 8 bits are needed.
        *
        * Make sure to keep the lowest 2 bits of addr clear for AD8400!
        */
       *(uint8_t *)(spiTxBuf + spiTxEnd) = pl;
       *((uint8_t *)(spiTxBuf + spiTxEnd) + 1) = addr;//bits 8 and 9 must be 0 for AD8400!
       ++spiTxEnd;
       if (spiTxEnd == SPI_TX_BUF_SIZE)
       {
          spiTxEnd = 0;//this is a circular buffer; wrap back around
       }
    } else {
       //else discard to avoid overflow
       spawnTxThread = false;
    }
```

```
    END_ATOMIC;//end critical section

    if (spawnTxThread)
    {
        spiTxWorker();
    }
}



/*
 * timerDrv.c
 *
 * Author: Michael Reinhart Sandstedt
 *
 * First release: September 14th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"
#include "constants.h"
#include "commsDefines.h"
#include "fsmStates.h"
#include "uartDrv.h"
#include "timer.h"
#include "calculateVectors.h"

void timerInit(void)
{
    T1CONbits.TON = 0;// stop Timer1
    T1CONbits.TSIDL = 1; // Discontinue timer operation when device enters Idle mode
    T1CONbits.TGATE = 0; // Gated time accumulation disabled
    T1CONbits.TCKPS = 0b00; // 1:1 prescale value
    T1CONbits.TCS = 0; // Internal clock as source
    TMR1 = 0x00; // clear Timer1
    PR1 = 0xFFFF; // load an arbitrary value for comparison (not 0)
    IPC0bits.T1IP = 0x02; // Set Timer 1 Interrupt Priority Level
    IFS0bits.T1IF = 0; // Clear Timer 1 Interrupt Flag
    IEC0bits.T1IE = 1; // Enable Timer1 interrupt

    T3CONbits.TON = 0; // Stop any 16-bit Timer3 operation
    T2CONbits.TON = 0; // Stop any 16/32-bit Timer3 operation
    T2CONbits.T32 = 1; // Enable 32-bit Timer mode
    T2CONbits.TCS = 0; // Select internal instruction cycle clock
    T2CONbits.TGATE = 0; // Disable Gated Timer mode
    T2CONbits.TCKPS = 0b00; // Select 1:1 Prescaler
    TMR3 = 0x00; // Clear 32-bit Timer (msw)
    TMR2 = 0x00; // Clear 32-bit Timer (lsw)
    PR2 = 0xFFFF; // Timer 2/3 used for busy wait; no hardware reset
    PR3 = 0xFFFF; // Timer 2/3 used for busy wait; no hardware reset
    IFS0bits.T2IF = 0; // Clear Timer2 Interrupt Flag
    IFS0bits.T3IF = 0; // Clear Timer3 Interrupt Flag
    IEC0bits.T2IE = 0; // Disable Timer2 interrupt
    IEC0bits.T3IE = 0; // Disable Timer3 interrupt

}

void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt(void)
{
    IFS0bits.T1IF = 0;
    T1CONbits.TON = 0;
    TMR1 = 0x0000;
    calculateFinalVectors();
}

void busy_wait_ms(uint16_t ms)
{
    uint32_t countTarget = PROCESSOR_CYCLES_PER_MS * ms - 87;//subtract cycles to compensate for multiplication and
funciton call overhead
```

```
    uint32_t count;

    if (countTarget > 0)
    {
        TMR3 = 0x0000; // Clear 32-bit Timer (msw)
        TMR2 = 0x0000; // Clear 32-bit Timer (lsw)

        T2CONbits.TON = 1;
        T3CONbits.TON = 1;

        do {
            count = TMR2;
            *((__eds__ uint16_t *)&count + 1) = TMR3HLD;
        } while (count < countTarget);

        T2CONbits.TON = 0;
        T3CONbits.TON = 0;
    }
}



/*****************************************************************************
* © 2005 Microchip Technology Inc.
*
* FileName:        traps.c
* Dependencies:    Header (.h) files if applicable, see below
* Processor:       dsPIC33Exxxx/PIC24Exxxx
* Compiler:        MPLAB® C30 v3.30 or higher
*
* SOFTWARE LICENSE AGREEMENT:
* Microchip Technology Incorporated ("Microchip") retains all ownership and
* intellectual property rights in the code accompanying this message and in all
* derivatives hereto.  You may use this code, and any derivatives created by
* any person or entity by or on your behalf, exclusively with Microchip's
* proprietary products.  Your acceptance and/or use of this code constitutes
* agreement to the terms and conditions of this notice.
*
* CODE ACCOMPANYING THIS MESSAGE IS SUPPLIED BY MICROCHIP "AS IS".  NO
* WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
* TO, IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY AND FITNESS FOR A
* PARTICULAR PURPOSE APPLY TO THIS CODE, ITS INTERACTION WITH MICROCHIP'S
* PRODUCTS, COMBINATION WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
*
* YOU ACKNOWLEDGE AND AGREE THAT, IN NO EVENT, SHALL MICROCHIP BE LIABLE, WHETHER
* IN CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE OR BREACH OF STATUTORY DUTY),
* STRICT LIABILITY, INDEMNITY, CONTRIBUTION, OR OTHERWISE, FOR ANY INDIRECT, SPECIAL,
* PUNITIVE, EXEMPLARY, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, FOR COST OR EXPENSE OF
* ANY KIND WHATSOEVER RELATED TO THE CODE, HOWSOEVER CAUSED, EVEN IF MICROCHIP HAS BEEN
* ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE.  TO THE FULLEST EXTENT
* ALLOWABLE BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO
* THIS CODE, SHALL NOT EXCEED THE PRICE YOU PAID DIRECTLY TO MICROCHIP SPECIFICALLY TO
* HAVE THIS CODE DEVELOPED.
*
* You agree that you are solely responsible for testing the code and
* determining its suitability.  Microchip has no obligation to modify, test,
* certify, or support the code.
*
* REVISION HISTORY:
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Author            Date      Comments on this revision
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Settu D                     07/09/06  First release of source file
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Srikar D                    04/27/11  Updated for dsPIC33E / PIC24E
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*
* ADDITIONAL NOTES:
* 1. This file contains trap service routines (handlers) for hardware
*    exceptions generated by the dsPIC33E device.
* 2. All trap service routines in this file simply ensure that device
```

```
*    continuously executes code within the trap service routine. Users
*    may modify the basic framework provided here to suit to the needs
*    of their application.
*
*********************************************************************/

#include "spintronicsIncludes.h"

void __attribute__((__interrupt__)) _OscillatorFail(void);
void __attribute__((__interrupt__)) _AddressError(void);
void __attribute__((__interrupt__)) _StackError(void);
void __attribute__((__interrupt__)) _MathError(void);
void __attribute__((__interrupt__)) _DMACError(void);


/*
Primary Exception Vector handlers:
These routines are used if INTCON2bits.ALTIVT = 0.
All trap service routines in this file simply ensure that device
continuously executes code within the trap service routine. Users
may modify the basic framework provided here to suit to the needs
of their application.
*/
void __attribute__((interrupt, no_auto_psv)) _OscillatorFail(void)
{
   INTCON1bits.OSCFAIL = 0;        //Clear the trap flag
   while (1);
}

void __attribute__((interrupt, no_auto_psv)) _AddressError(void)
{
   INTCON1bits.ADDRERR = 0;        //Clear the trap flag
   while (1);
}
void __attribute__((interrupt, no_auto_psv)) _StackError(void)
{
   INTCON1bits.STKERR = 0;         //Clear the trap flag
   while (1);
}

void __attribute__((interrupt, no_auto_psv)) _MathError(void)
{
   INTCON1bits.MATHERR = 0;        //Clear the trap flag
   while (1);
}

void __attribute__((interrupt, no_auto_psv)) _DMACError(void)
{
   INTCON1bits.DMACERR = 0;        //Clear the trap flag
   while (1);
}


/*
 * i2sDrv.c
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 * and Samiha Sultana
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include <string.h>
#include "spintronicsIncludes.h"
#include "spintronicsConfig.h"
#include "constants.h"
#include "fsmStates.h"
```

```
#include "uartDrv.h"
#include "commsDefines.h"
#include "balanceBridge.h"
#include "uartDrv.h"
#include "spintronicsStructs.h"
#include "digiPotDrv.h"
#include "utility.h"

#define DEFAULT_SENSOR_ADDRESS 0x00//bit pattern to disable both MUXs
#define USB_TX_BUF_SIZE 256
#define BT_TX_BUF_SIZE 256
#define USB_RX_BUF_SZ MAX_RX_PAYLOAD_SIZE
#define BT_RX_BUF_SZ MAX_RX_PAYLOAD_SIZE

#define FTDI_RST_BAR PORTFbits.RF5
#define USB_5V_DETECT PORTBbits.RB12

//function prototypes
static float setVolume(uint8_t channel, float voltage);
static inline void send(uint8_t* array, uint8_t numBytes);
static inline void receive(bool rxFromUSB, uint8_t *array, uint16_t rxPointer,
                    uint8_t sizeOfPayload);
static inline void decodeStartCommand(uint8_t startpayload[],
                                uint8_t sizeOfPayload);
static void usbTxWorker(void);
static void btTxWorker(void);
static inline bool copyToUSBTxBuf(uint8_t *array, uint16_t numBytes);
static inline bool copyToBTTxBuf(uint8_t *array, uint16_t numBytes);
static inline void decodeBalanceBridgeCommand(uint8_t *payload,
                                    uint8_t sizeOfPayload);
static float byte_array_to_float(__eds__ uint8_t *ptr);
static void float_to_byte_array(float input, __eds__ uint8_t *ptr);

#if !defined(NO_GUI) && !defined(SIMULATION_MODE)
static inline void processStartCommand(float GUISpecifiedA1,
                                float GUISpecifiedF1,
                                float GUISpecifiedA2,
                                float GUISpecifiedF2,
                                float GUISpecifiedT,
                                uint8_t GUISpeciedBridgeGainFactor,
                                float GUISpecifiedBridgeAnalogGain);
#endif

//global variables
uint8_t global_state = IDLE;
uint16_t measurementTime;//units are samples * 1024
_Q31_t stg3_coef;//stg3 has measurementTime samples; mpy by stg3_coef for avg
_Q15 f1;//units are Q15 half-cycles per sample-period
_Q15 f2;//units are Q15 half-cycles per sample-period
_Q15 fdiff;//units are Q15 half-cycles per sample-period
_Q15 fsum;//units are Q15 half-cycles per sample-period
_Q15 a1;//units are Q15 franctions of DAC full-scale
_Q15 a2;//units are Q15 fractions of DAC full-scale
uint8_t bridgeADCGainFactor;//this can be 0, 1, 2, 3, 4, 5, 6, 7, 8; gain = 2^bridgeADCGainFactor;//in practice, only 0, 1, 2, 3, 4 offer
any advantage due to the noise floor of the ADC (~114dB for CS4272)
uint8_t u24_code;
float inverseBridgeAnalogGain;
bool f1PlusF2OutOfRange;
_Q15 bridge_balance_amplitude;
_Q15 bridge_balance_frequency;

uint8_t *sensorAddressTable;
uint8_t sensorAddressTableAllocated[MAX_MUX_ADDRESS_TABLE_SIZE];
uint8_t numberOfSensors;

static uint8_t usbTxBuf [USB_TX_BUF_SIZE];//must be global so as to be accessible from an ISR//static means FILE SCOPE
ONLY
static uint8_t btTxBuf [BT_TX_BUF_SIZE];//must be global so as to be accessible from an ISR//static means FILE SCOPE ONLY
static int16_t usbTxCur, usbTxEnd, btTxCur, btTxEnd;//must be global so as to be accessible from an ISR//static means FILE SCOPE
ONLY//keep it singed so my head doesn't hurt!
```

```
#if !defined (NO_GUI) && !defined(SIMULATION_MODE)
static
#endif
inline void
processStartCommand(float GUISpecifiedA1, float GUISpecifiedF1,
                    float GUISpecifiedA2, float GUISpecifiedF2,
                    float GUISpecifiedT, uint8_t GUISpeciedBridgeGainFactor,
                    float GUISpecifiedBridgeAnalogGain) {

   int32_t local_T;//units are samples
   uint8_t T_high_bit_detect;
   _Q15 local_f1, local_f2, local_fdiff, local_fsum, local_bridge_balance_frequency;//units are Q15 half-cycles per sample-period
   _Q15 local_a1, local_a2, local_bridge_balance_amplitude;//units are Q15 franctions of DAC full-scale
   uint8_t local_gain_factor;//this can be 0, 1, 2, 3, 4, 5, 6, 7, 8; gain = 2^bridgeADCGainFactor;//in practice, only 0, 1, 2, 3, 4 offer
any advantage due to the noise floor of the ADC (~114dB for CS4272)
   uint8_t local_u24_code;
   float local_inverseBridgeAnalogGain;
   float implementedBridgeGain;
   bool local_f1PlusF2OutOfRange;

   float implementedF1, implementedF2, implementedFSum, implementedFDiff;
   float implementedA1, implementedA2, implementedT, maxMeasurementSamples, minMeasurementSamples;
   uint8_t implementedBridgeGainFactor, startPayload_confirmToGUI[28], i;
   float tempTime;

   if (GUISpecifiedA1 > FULLSCALE_BRIDGE_DAC_VOLTAGE) {

      transmitError(A1_OUT_OF_RANGE);
      local_a1 = 0x7FFF;
      implementedA1 = FULLSCALE_BRIDGE_DAC_VOLTAGE;
      local_bridge_balance_amplitude = DEFAULT_BALANCE_AMPLITUDE;//set in case we need to balance the bridge

   } else if (GUISpecifiedA1 < 0.0) {

      transmitError(A1_OUT_OF_RANGE);
      local_a1 = 0x0000;
      implementedA1 = 0.0;
      local_bridge_balance_amplitude = DEFAULT_BALANCE_AMPLITUDE;//set in case we need to balance the bridge

   } else {
      local_a1 = _Q15ftoi(GUISpecifiedA1 / FULLSCALE_BRIDGE_DAC_VOLTAGE);
      implementedA1 = _itofQ15(local_a1) * FULLSCALE_BRIDGE_DAC_VOLTAGE;
      local_bridge_balance_amplitude = local_a1;//set in case we need to balance the bridge
   }

   if (GUISpecifiedA2 > FULLSCALE_COIL_DAC_VOLTAGE) {

      transmitError(A2_OUT_OF_RANGE);
      local_a2 = 0x7FFF;
      implementedA2 = FULLSCALE_COIL_DAC_VOLTAGE;

   } else if (GUISpecifiedA2 < 0.0) {

      transmitError(A2_OUT_OF_RANGE);
      local_a2 = 0x0000;
      implementedA2 = 0.0;

   } else {

      local_a2 = _Q15ftoi(GUISpecifiedA2 / FULLSCALE_COIL_DAC_VOLTAGE);
      implementedA2 = _itofQ15(local_a2) * FULLSCALE_COIL_DAC_VOLTAGE;

   }

   maxMeasurementSamples = MAX_MEASUREMENT_SAMPLES;
   minMeasurementSamples = MIN_MEASUREMENT_SAMPLES;

   if (GUISpecifiedF1 > MAX_OUTPUT_HZ) {
      transmitError(F1_OUT_OF_RANGE);
      local_f1 = _Q15ftoi(MAX_OUTPUT_HZ * TWICE_SAMPLE_PERIOD);
      local_bridge_balance_frequency = DEFAULT_BALANCE_FREQUENCY;//set in case we need to balance the bridge
```

```
    } else if (GUISpecifiedF1 < 0) {
       transmitError(F1_OUT_OF_RANGE);
       local_f1 = 0;
       local_bridge_balance_frequency = DEFAULT_BALANCE_FREQUENCY;//set in case we need to balance the bridge
    } else {
       local_f1 = _Q15ftoi(GUISpecifiedF1 * TWICE_SAMPLE_PERIOD);
       local_bridge_balance_frequency = local_f1;
    }

    if (GUISpecifiedF2 > MAX_OUTPUT_HZ) {
       transmitError(F2_OUT_OF_RANGE);
       local_f2 = _Q15ftoi(MAX_OUTPUT_HZ * TWICE_SAMPLE_PERIOD);
    } else if (GUISpecifiedF2 < 0) {
       transmitError(F2_OUT_OF_RANGE);
       local_f2 = 0;
    } else {
       local_f2 = _Q15ftoi(GUISpecifiedF2 * TWICE_SAMPLE_PERIOD);
    }

    tempTime = GUISpecifiedT * SAMPLE_RATE;
    if (tempTime > maxMeasurementSamples) {
       transmitError(T_OUT_OF_RANGE);
       local_T = MAX_MEASUREMENT_SAMPLES;
    } else if (tempTime < minMeasurementSamples) {
       transmitError(T_OUT_OF_RANGE);
       local_T = MIN_MEASUREMENT_SAMPLES;
    } else {
       local_T = (int32_t)tempTime;
    }
    //making local_T divisible by 4096 keeps f1 and f2 close to user values
    local_T &= 0xFFFFF000;

    //get local_T as close as possible
    if ((int32_t)tempTime - local_T > 0x00000800) {
       local_T += 0x00001000;
    }

    {
       // calculate magic values for f2, f1

       #define TWO_TO_N 65536 // our trig arguments are 16bit
       #define MAX_MOD 0 // 0 ensures perfect phase alignment of all signals at T //max_mod = (int16_t)(max_phase_offset / 180.0 *
(1 << (n - 1)));//n=16

       int16_t f2_min, f2_max, f1_min, f1_max;
       bool success = false;
       int16_t max_delta = 0;
       int32_t mod_offset;

       f2_min = local_f2;
       f2_max = local_f2;
       f1_min = local_f1 - 1;
       f1_max = local_f1 + 1;

       while (!success) {
          f2_max += max_delta;
          if (f2_max >= MAX_OUTPUT_FREQUENCY) {
             f2_max = MAX_OUTPUT_FREQUENCY;
          }
          f1_min -= max_delta;
          if (f1_min  <= 0) {
             f1_min = 0;
          }
          f1_max += max_delta;
          if (f1_max >= MAX_OUTPUT_FREQUENCY) {
             f1_max = MAX_OUTPUT_FREQUENCY;
          }

          for (local_f2 = f2_min; local_f2 <= f2_max; ++local_f2) {
             if (local_T * (int32_t)local_f2 % TWO_TO_N == 0) {
```

```
            for (local_f1 = f1_min; f1 <= f1_max; ++local_f1) {
               mod_offset = local_T * (int32_t)local_f1 % TWO_TO_N;
               if (mod_offset <= MAX_MOD) {
                  success = true;
                  break;
               }
            }
         }
         if (success) {
            break;
         }
         ++max_delta;
      }

      f2_min -= max_delta;
      if (f2_min  <= 0) {
         f2_min = 0;
      }
      if (   f1_min == 0
         && f2_min == 0
         && f1_max == MAX_OUTPUT_FREQUENCY
         && f1_max == MAX_OUTPUT_FREQUENCY
         && !success) {
         transmitError(F1_OUT_OF_RANGE);
         transmitError(F2_OUT_OF_RANGE);
         transmitError(T_OUT_OF_RANGE);
         return;
      }
   }
}

implementedF2 = _itofQ15(local_f2) * HALF_SAMPLE_RATE;
implementedF1 = _itofQ15(local_f1) * HALF_SAMPLE_RATE;
implementedT = (float)local_T / SAMPLE_RATE;

local_f1PlusF2OutOfRange = false;
if (GUISpecifiedF1 + GUISpecifiedF2 > MAX_OUTPUT_HZ) {
   local_f1PlusF2OutOfRange = true;
   transmitError(F1_PLUS_F2_OUT_OF_RANGE);
   local_fsum = 0;
   implementedFSum = 0;
} else {
   local_fsum = local_f1 + local_f2;
   implementedFSum = implementedF1 + implementedF2;
}
local_fdiff = _Q15abs(local_f1 - local_f2);
implementedFDiff = abs(implementedF1 - implementedF2);

switch(GUISpeciedBridgeGainFactor)
{
   case 1:
      local_gain_factor = 0;
      implementedBridgeGainFactor = 1;
      break;
   case 2:
      local_gain_factor = 1;
      implementedBridgeGainFactor = 2;
      break;
   case 4:
      local_gain_factor = 2;
      implementedBridgeGainFactor = 4;
      break;
   case 8:
      local_gain_factor = 3;
      implementedBridgeGainFactor = 8;
      break;
   case 16:
      local_gain_factor = 4;
      implementedBridgeGainFactor = 16;
      break;
   default:
```

```
            transmitError(INVALID_DIGITAL_GAIN_VALUE);
            local_gain_factor = 0;
            implementedBridgeGainFactor = 1;
            break;
    }

#ifdef FIXED_ANALOG_GAIN
    GUISpecifiedBridgeAnalogGain = FIXED_ANALOG_GAIN_VALUE;
#endif

    if (GUISpecifiedBridgeAnalogGain < BRIDGE_ADC_BUFFER_MIN_GAIN) {
        transmitError(ANALOG_GAIN_OUT_OF_RANGE);
        local_u24_code = 0x00;
        implementedBridgeGain = BRIDGE_ADC_BUFFER_MIN_GAIN;
        local_inverseBridgeAnalogGain = INVERSE_BRIDGE_ANALOG_MIN_GAIN;
    } else if (GUISpecifiedBridgeAnalogGain > BRIDGE_ADC_BUFFER_MAX_GAIN) {
        transmitError(ANALOG_GAIN_OUT_OF_RANGE);
        local_u24_code = 0xFF;
        implementedBridgeGain = BRIDGE_ADC_BUFFER_MAX_GAIN;
        local_inverseBridgeAnalogGain = INVERSE_BRIDGE_ANALOG_MAX_GAIN;
    } else {
        local_u24_code = getU24CodeFromBrdigeBufGain(GUISpecifiedBridgeAnalogGain);
        implementedBridgeGain = getBridgeBufGainFromU24Code(local_u24_code);
        local_inverseBridgeAnalogGain = getBridgeInverseGainFromU24Code(local_u24_code);
    }

    startPayload_confirmToGUI[0] = CONFIRM_START_COMMAND;
    startPayload_confirmToGUI[1] = 0x19;
    float_to_byte_array(implementedA1, &startPayload_confirmToGUI[2]);
    float_to_byte_array(implementedF1, &startPayload_confirmToGUI[6]);
    float_to_byte_array(implementedA2, &startPayload_confirmToGUI[10]);
    float_to_byte_array(implementedF2, &startPayload_confirmToGUI[14]);
    float_to_byte_array(implementedT, &startPayload_confirmToGUI[18]);
    float_to_byte_array(implementedBridgeGain, &startPayload_confirmToGUI[22]);
    startPayload_confirmToGUI[26] = implementedBridgeGainFactor;

    startPayload_confirmToGUI[27] = 0;
    for (i = 0; i < 27; i++)
    {
        startPayload_confirmToGUI[27] = startPayload_confirmToGUI[27] ^ startPayload_confirmToGUI[i];
    }
    send(startPayload_confirmToGUI, 28);

    START_ATOMIC();//begin critical section; must be atomic!
    measurementTime = (local_T >> 10) & 0xFFFF;//need to divide by 1024
    stg3_coef.Q31 = 0x7FFFFFFF / measurementTime;
    f1 = local_f1;
    f2 = local_f2;
    fdiff = local_fdiff;
    fsum = local_fsum;
    a1 = local_a1;
    a2 = local_a2;
    bridgeADCGainFactor = local_gain_factor;
    u24_code = local_u24_code;
    inverseBridgeAnalogGain = local_inverseBridgeAnalogGain;
    f1PlusF2OutOfRange = local_f1PlusF2OutOfRange;
    bridge_balance_amplitude = local_bridge_balance_amplitude;
    bridge_balance_frequency = local_bridge_balance_frequency;
    global_state = RAMP_DOWN_COIL_RESTART;
    END_ATOMIC();//end critical section

}

void uart_Init (void)
{
    uint8_t junk;

    /*****************************************************************************
     * initialization of global variables
     *****************************************************************************/
```

```
    sensorAddressTable = sensorAddressTableAllocated;
    measurementTime = SAMPLE_RATE;
    f1 = 0;
    f2 = 0;
    fdiff = 0;
    fsum = 0;
    bridgeADCGainFactor = 1;
    f1PlusF2OutOfRange = false;
    a1 = 0x7FFF;
    a2 = 0x7FFF;
    numberOfSensors = 1;
    sensorAddressTable[0] = DEFAULT_SENSOR_ADDRESS;
    usbTxCur = 0;
    usbTxEnd = 0;
    btTxCur = 0;
    btTxEnd = 0;


    /*****************************************************************************
     * pin setup
     *****************************************************************************/

    //RP101/RF5/PIN32 connects to USB_RESET via Si8442
    TRISFbits.TRISF5 = 0;//set RF5 to be an output so we can toggle the reset pin of the FTDI via the Si8442
    FTDI_RST_BAR = 0;//bring reset low (active)

    //RPI44/RB12/PIN27 connects to USB5V via Si8442
    ANSELBbits.ANSB12 = 0;//make RB12 digital
    TRISBbits.TRISB12 = 1;//set RB12 to be an input so we can monitor whether USB is plugged in or not; RB12 is connected to
USB5V via the Si8442

    //RP100/RF4/PIN31 connects to RX_BT
    //PORTF is digital by default
    RPOR8bits.RP99R = 0x03;//route U2TX to RP99

    //RPI46/RB14/PIN29 connects to TX_BT
    ANSELBbits.ANSB14 = 0;//make RB14 digital
    RPINR19bits.U2RXR = 46;//route RPI46 to U2RXR

    //RPI47/RB15/PIN30 connects to FTDI_TX via Si8442
    ANSELBbits.ANSB15 = 0;//make RB15 digital
    RPINR18bits.U1RXR = 47;//route RPI47 to U1RXR

    //RP100/RF4/PIN31 connects to FTDI_RX via Si8442
    RPOR9bits.RP100R = 0x01;//route U1TX to RP100


    /*****************************************************************************
     * UART1 setup for connection to USB via FTDI
     *****************************************************************************/

    U1MODEbits.USIDL = 0;
    U1MODEbits.IREN = 0;
    U1MODEbits.RTSMD = 1;     //U1RTS in Simplex mode
    U1MODEbits.UEN = 0;
    U1MODEbits.WAKE = 0;
    U1MODEbits.ABAUD = 0;
    U1MODEbits.URXINV = 0;
    U1MODEbits.BRGH = 0;                    //Set for slow speed mode
    U1MODEbits.PDSEL = 0;      //8-bit mode, no parity
    U1MODEbits.STSEL = 0;      //1-stop bit
    U1MODEbits.UARTEN = 1;   //UART1 is enabled
    U1MODEbits.LPBACK = 0;   //LPBACK Disabled

    /****************************** Baud rate Calculations**********************
    *
    *   U1BRG= [(Fcy/(Desired_Baud_rate*16)] - 1....Provided BRGH= 0 (Slow mode)
    *   U1BRG= [Fcy/(4*Baud_rate)] - 1.............Provided BRGH= 1 (Fast mode)
    *   Fosc= 140Mhz
    *   Fcy= Fosc/2= 70MHz
    *
```

```
   ********************************************************************************/
   U1BRG = 433;              //Set for baudrate of 10081
   //U1BRG = 437;            //Set for baudrate of 10000 Baud
   //U1BRG = 37;             //Set for baudrate of 115200 baud
   //U1BRG = 455;            //Set for baudrate of 9600 baud
   //U1BRG = 451;            //Set for baudrate of 9677

   IPC3bits.U1TXIP = 4;       //TX interrupt priority
   U1STAbits.UTXISEL0 = 1;    //Interrupt when the last character is shifted out of the Transmit Shift Register and all transmit
operations are completed
   U1STAbits.UTXISEL1 = 0;
   IFS0bits.U1TXIF = 0;       //clear TX interrupt flag
   IEC0bits.U1TXIE = 0;       //enable UART TX Interrupt
   U1STAbits.UTXEN = 1;       //enable UART.TXEN

   IEC0bits.U1RXIE = 0;       //disable RX interrupt
   IPC2bits.U1RXIP = 3;          //Receive interrupt priority
   U1STAbits.URXISEL = 0;     //Interrupt is set when any character is received and transferred from the UxRSR to the receive
buffer; receive buffer has one or more characters


   /********************************************************************************
    * UART2 setup for connection to blue tooth
    ********************************************************************************/

   U2MODEbits.USIDL = 0;
   U2MODEbits.IREN = 0;
   U2MODEbits.RTSMD = 1;     //U2RTS in Simplex mode
   U2MODEbits.UEN = 0;
   U2MODEbits.WAKE = 0;
   U2MODEbits.ABAUD = 0;
   U2MODEbits.URXINV = 0;
   U2MODEbits.BRGH = 0;                  //Set for slow speed mode
   U2MODEbits.PDSEL = 0;      //8-bit mode, no parity
   U2MODEbits.STSEL = 0;      //1-stop bit
   U2MODEbits.UARTEN = 1;   //UART2 is enabled
   U2MODEbits.LPBACK = 0;   //LPBACK Disabled

   U2BRG = 37;               //Set for baudrate of 115200 baud

   IPC7bits.U2TXIP = 4;       //TX interrupt priority
   U2STAbits.UTXISEL0 = 1;    //Interrupt when the last character is shifted out of the Transmit Shift Register; all transmit
operations are completed
   U2STAbits.UTXISEL1 = 0;
   IFS1bits.U2TXIF = 0;
   IEC1bits.U2TXIE = 0;
   U2STAbits.UTXEN = 1;       //enable UART.TXEN

   IEC1bits.U2RXIE = 0;       //disable RX interrupt
   IPC7bits.U2RXIP = 3;          //Receive interrupt priority
   U2STAbits.URXISEL = 0;


   /********************************************************************************
    * clear interrupt flags, clear buffers, enable interrupts
    ********************************************************************************/

#ifndef SIMULATION_MODE
   busy_wait_ms(10);          //wait for the FTDI to reset
   FTDI_RST_BAR = 1;          //start the FTDI
   busy_wait_ms(50);          //wait for the FTDI to start
#endif

   IFS0bits.U1RXIF = 0;       //clear U1RXIF
   U1STAbits.OERR = 0;        //clear overflow error flag
   while (1 == U1STAbits.URXDA) {
      junk = U1RXREG;//clear the input buffer
   }
   IEC0bits.U1RXIE = 1;       //enable RX interrupt

   IFS1bits.U2RXIF = 0;       //clear U2RXIF
```

```
    U2STAbits.OERR = 0;         //clear overflow error flag
    while (1 == U2STAbits.URXDA) {
        junk = U2RXREG;//clear the input buffer
    }
    IEC1bits.U2RXIE = 1;        //enable RX interrupt
}

void __attribute__((__interrupt__, no_auto_psv)) _U1RXInterrupt(void)
{
    static uint8_t rxState = RX_IDLE;
    static uint16_t endDataRxPointer = 0;
    static uint16_t numBytesInPayload = 0;
    static uint8_t USB_RxBuffer[USB_RX_BUF_SZ] ={0};
    uint8_t rxByte;

    IFS0bits.U1RXIF = 0;

    if(0 == USB_5V_DETECT) {

        if (1 == U1STAbits.OERR) {
            U1STAbits.OERR = 0;//this also clears U1RXREG
        } else {
            while (1 == U1STAbits.URXDA) {
                rxByte = U1RXREG;//clear the input buffer
            }
        }

        rxState = RX_IDLE;
        numBytesInPayload = 0;

        /*
         * USB is not connected;
         * Return after the input buffer and counters are cleared.
         */
        return;
    }

    if (1 == U1STAbits.OERR) {

        U1STAbits.OERR = 0;//this also clears U1RXREG

        rxState = RX_IDLE;
        numBytesInPayload = 0;

        transmitError(UART_RX_BUFFER_OVERFLOW);

        /*
         * An overflow has occurred
         * Return after the input buffer and counters are cleared.
         */
        return;
    }

    while (U1STAbits.URXDA) {

        rxByte = U1RXREG;

        switch (rxState) {
            case RX_IDLE:
                if (START_FLAG == rxByte) {
                    rxState = RX_MID_PAYLOAD;
                    numBytesInPayload = 0;
                }
                break;
            case RX_MID_PAYLOAD:
                if (ESCAPE_FLAG == rxByte) {
                    rxState = RX_ESCAPE;
                } else if (STOP_FLAG == rxByte) {
                    if (numBytesInPayload > (endDataRxPointer + 1)) {
                        receive(true, USB_RxBuffer, endDataRxPointer + USB_RX_BUF_SZ - numBytesInPayload, numBytesInPayload);
                    } else {
```

```
                receive(true, USB_RxBuffer, endDataRxPointer - numBytesInPayload, numBytesInPayload);
            }
            rxState = RX_IDLE;
          } else if (START_FLAG == rxByte) {
            transmitError(MALFORMED_PACKET_RECEIVED);
            rxState = RX_MID_PAYLOAD;
            numBytesInPayload = 0;
          } else {
            USB_RxBuffer[endDataRxPointer] = rxByte;
            ++endDataRxPointer;
            ++numBytesInPayload;
            if (USB_RX_BUF_SZ == endDataRxPointer) {
              endDataRxPointer = 0;
            }
          }
          break;
        case RX_ESCAPE:
          if (START_FLAG == rxByte) {
            transmitError(MALFORMED_PACKET_RECEIVED);
            rxState = RX_MID_PAYLOAD;
            numBytesInPayload = 0;
            break;
          }

          rxByte ^= ESCAPE_XOR;

          if (START_FLAG == rxByte || ESCAPE_FLAG == rxByte || STOP_FLAG == rxByte) {
            USB_RxBuffer[endDataRxPointer] = rxByte;
            ++endDataRxPointer;
            ++numBytesInPayload;
            if (USB_RX_BUF_SZ == endDataRxPointer) {
              endDataRxPointer = 0;
            }
            rxState = RX_MID_PAYLOAD;
          } else {
            transmitError(MALFORMED_PACKET_RECEIVED);
            rxState = RX_IDLE;
          }
          break;
        default:
          rxState = RX_IDLE;
          break;
      }

      if (1 == U1STAbits.OERR) {

        U1STAbits.OERR = 0;//this also clears U1RXREG

        rxState = RX_IDLE;
        numBytesInPayload = 0;

        transmitError(UART_RX_BUFFER_OVERFLOW);

        /*
         * An overflow has occurred
         * Return after the input buffer and counters are cleared.
         */
        return;
      }
    }
}

void __attribute__((__interrupt__, no_auto_psv)) _U2RXInterrupt(void)
{
    static uint8_t rxState = RX_IDLE;
    static uint16_t endDataRxPointer = 0;
    static uint16_t numBytesInPayload = 0;
    static uint8_t BT_RxBuffer[BT_RX_BUF_SZ] ={0};
    uint8_t rxByte;

    IFS1bits.U2RXIF = 0;
```

```
if(1 == USB_5V_DETECT) {

   if (1 == U2STAbits.OERR) {
      U2STAbits.OERR = 0;//this also clears U1RXREG
   } else {
      while (1 == U2STAbits.URXDA) {
         rxByte = U2RXREG;//clear the input buffer
      }
   }

   rxState = RX_IDLE;
   numBytesInPayload = 0;

   /*
    * USB is connected;
    * Return after the input buffer and counters are cleared.
    */
   return;
}

if (1 == U2STAbits.OERR) {

   U2STAbits.OERR = 0;//this also clears U2RXREG

   rxState = RX_IDLE;
   numBytesInPayload = 0;

   transmitError(UART_RX_BUFFER_OVERFLOW);

   /*
    * An overflow has occurred
    * Return after the input buffer and counters are cleared.
    */
   return;
}

while (U2STAbits.URXDA) {

   rxByte = U2RXREG;

   switch (rxState) {
      case RX_IDLE:
         if (START_FLAG == rxByte) {
            rxState = RX_MID_PAYLOAD;
            numBytesInPayload = 0;
         }
         break;
      case RX_MID_PAYLOAD:
         if (ESCAPE_FLAG == rxByte) {
            rxState = RX_ESCAPE;
         } else if (STOP_FLAG == rxByte) {
            if (numBytesInPayload > (endDataRxPointer + 1)) {
               receive(false, BT_RxBuffer, endDataRxPointer + BT_RX_BUF_SZ - numBytesInPayload, numBytesInPayload);
            } else {
               receive(false, BT_RxBuffer, endDataRxPointer - numBytesInPayload, numBytesInPayload);
            }
            rxState = RX_IDLE;
         } else if (START_FLAG == rxByte) {
            transmitError(MALFORMED_PACKET_RECEIVED);
            rxState = RX_MID_PAYLOAD;
            numBytesInPayload = 0;
         } else {
            BT_RxBuffer[endDataRxPointer] = rxByte;
            ++endDataRxPointer;
            ++numBytesInPayload;
            if (BT_RX_BUF_SZ == endDataRxPointer) {
               endDataRxPointer = 0;
            }
         }
         break;
```

```
            case RX_ESCAPE:
               if (START_FLAG == rxByte) {
                  transmitError(MALFORMED_PACKET_RECEIVED);
                  rxState = RX_MID_PAYLOAD;
                  numBytesInPayload = 0;
                  break;
               }

               rxByte ^= ESCAPE_XOR;

               if (START_FLAG == rxByte || ESCAPE_FLAG == rxByte || STOP_FLAG == rxByte) {
                  BT_RxBuffer[endDataRxPointer] = rxByte;
                  ++endDataRxPointer;
                  ++numBytesInPayload;
                  if (BT_RX_BUF_SZ == endDataRxPointer) {
                     endDataRxPointer = 0;
                  }
                  rxState = RX_MID_PAYLOAD;
               } else {
                  transmitError(MALFORMED_PACKET_RECEIVED);
                  rxState = RX_IDLE;
               }
               break;
            default:
               rxState = RX_IDLE;
               break;
         }

         if (1 == U2STAbits.OERR) {

            U2STAbits.OERR = 0;//this also clears U2RXREG

            rxState = RX_IDLE;
            numBytesInPayload = 0;

            transmitError(UART_RX_BUFFER_OVERFLOW);

            /*
             * An overflow has occurred
             * Return after the input buffer and counters are cleared.
             */
            return;
         }
      }
   }

   void __attribute__((__interrupt__, no_auto_psv)) _U1TXInterrupt(void)
   {
      IFS0bits.U1TXIF = 0;
      IEC0bits.U1TXIE = 0;
      usbTxWorker();//The shift register got full. But now it's empty again.  Let's TX!
   }

   void __attribute__((__interrupt__, no_auto_psv)) _U2TXInterrupt(void)
   {
      IFS1bits.U2TXIF = 0;
      IEC1bits.U2TXIE = 0;
      btTxWorker();//The shift register got full. But now it's empty again.  Let's TX!
   }

   inline void
   transmitResults(uint8_t sensor, __eds__ float_array_t *phaseAngle,
               __eds__ float_array_t *amplitude, bool bridgeADCClip,
               bool coilADCClip, bool bridgeDigitalClip)
   {
      uint8_t txbuffer[44],i;

      txbuffer[0] = REPORT_VALUES;
      txbuffer[1] = 0x29;
      txbuffer[2] = sensor;
```

```
#if !defined(MESURE_F2_AT_BRIDGE) && !defined(MESURE_F2_AT_COIL)
   float ZERO = 0.0;
#endif
   float_to_byte_array(amplitude->bridge_f1, &txbuffer[3]);
   float_to_byte_array(phaseAngle->bridge_f1, &txbuffer[7]);
#ifdef MEASURE_F2_AT_BRIDGE
   float_to_byte_array(amplitude->bridge_f2, &txbuffer[11]);
   float_to_byte_array(phaseAngle->bridge_f2, &txbuffer[15]);
#else
   float_to_byte_array(ZERO, &txbuffer[11]);
   float_to_byte_array(ZERO, &txbuffer[15]);
#endif
   float_to_byte_array(amplitude->bridge_fdiff, &txbuffer[19]);
   float_to_byte_array(phaseAngle->bridge_fdiff, &txbuffer[23]);
   float_to_byte_array(amplitude->bridge_fsum, &txbuffer[27]);
   float_to_byte_array(phaseAngle->bridge_fsum, &txbuffer[31]);
#ifdef MEASURE_F2_AT_COIL
   float_to_byte_array(amplitude->coil_f2, &txbuffer[35]);
   float_to_byte_array(phaseAngle->coil_f2, &txbuffer[39]);
#else
   float_to_byte_array(ZERO, &txbuffer[35]);
   float_to_byte_array(ZERO, &txbuffer[39]);
#endif

   txbuffer[43] =0;
   for (i = 0; i < 43; i++) {
      txbuffer[43] = txbuffer[43] ^ txbuffer[i];
   }
   send (txbuffer, 44);

   if (bridgeADCClip) {
      transmitError(BRIDGE_ADC_CLIP);
   }
   if (coilADCClip) {
      transmitError(COIL_ADC_CLIP);
   }
   if (bridgeDigitalClip) {
      transmitError(BRIDGE_DIGITAL_CLIP);
   }
}

inline void
transmitError(uint8_t errorCode)
{
   static uint8_t errorpayload[4];
   errorpayload[0] = REPORT_ERROR;
   errorpayload[1] = 0x01;
   errorpayload[2] = errorCode;
   errorpayload[3] = errorpayload[0] ^ errorpayload[1] ^ errorpayload[2];
   send(errorpayload, 4);
}

static inline void
send(uint8_t *array, uint8_t numBytes)
{
   if (1 == PORTBbits.RB12)//USB is connected; use USB for transmission
   {
      if (true == copyToUSBTxBuf(array, numBytes))
      {
         usbTxWorker();
      }
      //else: another thread is handling transmission, or a buffer overflow occurred
   }
   else
   {
      if (true == copyToBTTxBuf(array, numBytes))
      {
         btTxWorker();
      }
      //else: another thread is handling transmission, or a buffer overflow occurred
   }
```

```
}

static void
usbTxWorker(void)
{
   START_ATOMIC();//begin critical section; must be atomic!
   /* continue if there is data to transmit and the U2TXREG isn't full */
   while (usbTxEnd != usbTxCur && U1STAbits.UTXBF == 0)
   {
      U1TXREG = usbTxBuf[usbTxCur];
      ++usbTxCur;
      if (USB_TX_BUF_SIZE == usbTxCur)
      {
         usbTxCur = 0;
      }
   }
   if (usbTxEnd != usbTxCur) {
      IEC0bits.U1TXIE = 1;//enable interrupt to call the worker again
   }
   END_ATOMIC();//end critical section
}

static void
btTxWorker(void)
{
   START_ATOMIC();//begin critical section; must be atomic!
   /* continue if there is data to transmit and the U2TXREG isn't full */
   while (btTxEnd != btTxCur && U2STAbits.UTXBF == 0)
   {
      U2TXREG = btTxBuf[btTxCur];
      ++btTxCur;
      if (BT_TX_BUF_SIZE == btTxCur)
      {
         btTxCur = 0;
      }
   }
   if (btTxEnd != btTxCur) {
      IEC1bits.U2TXIE = 1;//enable interrupt to call the worker again
   }
   END_ATOMIC();//end critical section
}

static inline bool
copyToUSBTxBuf(uint8_t *array, uint16_t numBytes)
{
   bool spawnTxThread = false;
   uint16_t bufSpaceAvl, i;

   START_ATOMIC();//begin critical section; must be atomic!

   if (usbTxEnd == usbTxCur)
   {
      spawnTxThread = true;
      bufSpaceAvl = USB_TX_BUF_SIZE - 1;
   }
   else if (usbTxEnd > usbTxCur)
   {
      bufSpaceAvl = USB_TX_BUF_SIZE - 1 - (usbTxEnd - usbTxCur);
   }
   else
   {
      bufSpaceAvl = usbTxCur - usbTxEnd - 1;
   }

   if (bufSpaceAvl < numBytes * 2 + 2)//encoded payload is 2x the size with all escaped characters, + 2 for START and STOP flag
   {
      spawnTxThread = false;//no message transmitted; a buffer overflow may occur
   }
   else
   {
      usbTxBuf[usbTxEnd] = START_FLAG;
```

```
      ++usbTxEnd;
      if (usbTxEnd == USB_TX_BUF_SIZE)
      {
         usbTxEnd = 0;//this is a circular buffer; wrap back around
      }
      for (i = 0; i < numBytes; ++i)
      {
         if (array[i] == START_FLAG || array[i] == ESCAPE_FLAG || array[i] == STOP_FLAG) {
            usbTxBuf[usbTxEnd] = ESCAPE_FLAG;
            ++usbTxEnd;
            if (usbTxEnd == USB_TX_BUF_SIZE)
            {
               usbTxEnd = 0;//this is a circular buffer; wrap back around
            }
            usbTxBuf[usbTxEnd] = array[i] ^ ESCAPE_XOR;
            ++usbTxEnd;
            if (usbTxEnd == USB_TX_BUF_SIZE)
            {
               usbTxEnd = 0;//this is a circular buffer; wrap back around
            }
         } else {
            usbTxBuf[usbTxEnd] = array[i];
            ++usbTxEnd;
            if (usbTxEnd == USB_TX_BUF_SIZE)
            {
               usbTxEnd = 0;//this is a circular buffer; wrap back around
            }
         }
      }
      usbTxBuf[usbTxEnd] = STOP_FLAG;
      ++usbTxEnd;
      if (usbTxEnd == USB_TX_BUF_SIZE)
      {
         usbTxEnd = 0;//this is a circular buffer; wrap back around
      }
   }
   END_ATOMIC();//end critical section
   return spawnTxThread;
}

static inline bool
copyToBTTxBuf(uint8_t *array, uint16_t numBytes)
{
   bool spawnTxThread = false;
   uint16_t bufSpaceAvl, i;

   START_ATOMIC();//begin critical section; must be atomic!

   if (btTxEnd == btTxCur)
   {
      spawnTxThread = true;
      bufSpaceAvl = BT_TX_BUF_SIZE - 1;
   }
   else if (btTxEnd > btTxCur)
   {
      bufSpaceAvl = BT_TX_BUF_SIZE - 1 - (btTxEnd - btTxCur);
   }
   else
   {
      bufSpaceAvl = btTxCur - btTxEnd - 1;
   }

   if (bufSpaceAvl < numBytes * 2 + 2)//encoded payload is 2x the size with all escaped characters, + 2 for START and STOP flag
   {
      spawnTxThread = false;//no message transmitted; a buffer overflow may occur
   }
   else
   {
      btTxBuf[btTxEnd] = START_FLAG;
      ++btTxEnd;
      if (btTxEnd == BT_TX_BUF_SIZE)
```

```
        {
            btTxEnd = 0;//this is a circular buffer; wrap back around
        }
        for (i = 0; i < numBytes; ++i)
        {
            if (array[i] == START_FLAG || array[i] == ESCAPE_FLAG || array[i] == STOP_FLAG) {
                btTxBuf[btTxEnd] = ESCAPE_FLAG;
                ++btTxEnd;
                if (btTxEnd == BT_TX_BUF_SIZE)
                {
                    btTxEnd = 0;//this is a circular buffer; wrap back around
                }
                btTxBuf[btTxEnd] = array[i] ^ ESCAPE_XOR;
                ++btTxEnd;
                if (btTxEnd == BT_TX_BUF_SIZE)
                {
                    btTxEnd = 0;//this is a circular buffer; wrap back around
                }
            } else {
                btTxBuf[btTxEnd] = array[i];
                ++btTxEnd;
                if (btTxEnd == BT_TX_BUF_SIZE)
                {
                    btTxEnd = 0;//this is a circular buffer; wrap back around
                }
            }
        }
        btTxBuf[btTxEnd] = STOP_FLAG;
        ++btTxEnd;
        if (btTxEnd == BT_TX_BUF_SIZE)
        {
            btTxEnd = 0;//this is a circular buffer; wrap back around
        }
    }
    END_ATOMIC();//end critical section
    return spawnTxThread;
}

static inline void
receive (bool rxFromUSB, uint8_t *array, uint16_t rxPointer,
         uint8_t sizeOfPayload)
{
    uint16_t rx_buf_sz;
    uint8_t i, payload[MAX_RX_PAYLOAD_SIZE] = {0}, xor_byte = 0;

    if (true == rxFromUSB) {
        rx_buf_sz = USB_RX_BUF_SZ;
    } else {
        rx_buf_sz = BT_RX_BUF_SZ;
    }

    for(i = 0; i < sizeOfPayload; i++) {
        if (rxPointer == rx_buf_sz) {
            rxPointer = 0;
        }
        payload[i] = array[rxPointer];
        ++rxPointer;
    }

    //check XOR
    for (i = 0; i < (sizeOfPayload - 1); i++) {

        xor_byte = xor_byte ^ payload[i];

    } if (xor_byte != payload[i]) {

        transmitError(Bad_Packet_XOR);

    } else {

        switch (payload[0]) {
```

```
case START_COMMAND:

   decodeStartCommand(payload, sizeOfPayload);
   break;

case STOP_COMMAND:

   if (STOP_PAYLOAD_SIZE != sizeOfPayload) {

      transmitError(RECEIVED_PACKET_WITH_INCORRECT_SIZE);

   } else {

      START_ATOMIC();//begin critical section; must be atomic!
      global_state = RAMP_DOWN_COIL_QUIT;
      END_ATOMIC();//end critical section

      payload[0] = CONFIRM_STOP_COMMAND;
      payload[1] = 0;
      payload[2] = payload[0] ^ payload[1];
      send(payload, 3);

   }
   break;

case CONFIG_MUX_ADDRESSING:
{
   uint8_t newNumSensors;
   bool copyInNewTable = false;

   if(sizeOfPayload > MAX_RX_PAYLOAD_SIZE) {

      transmitError(RECEIVED_PACKET_WITH_INCORRECT_SIZE);

   } else {

      START_ATOMIC();//begin critical section; must be atomic!

      /*
       * check to see if the new table is different than the one
       * in memory
       */
      newNumSensors = sizeOfPayload - 3;

      if (newNumSensors == 0) {

         numberOfSensors = 1;
         sensorAddressTable[0] = DEFAULT_SENSOR_ADDRESS;
         sensorRBridgeTableValid = false;

      } else if (newNumSensors != numberOfSensors) {

         copyInNewTable = true;
         sensorRBridgeTableValid = false;

      } else {

         for (i = 0; i < numberOfSensors; i++) {

            if (sensorAddressTable[i] != payload[2+i]) {

               copyInNewTable = true;
               sensorRBridgeTableValid = false;
               break;
            }

         }
      }

      if (copyInNewTable) {
```

```
                numberOfSensors = newNumSensors;

                for (i = 0; i < numberOfSensors; i++) {

                    sensorAddressTable[i] = payload[2+i];

                }
            }
            END_ATOMIC();//end critical section

            payload[0] = CONFIRM_MUX_ADDRESSING;
            payload[1] = 0;
            payload[2] = payload[0] ^ payload[1];
            send(payload, 3);
          }
          break;
        }
        case BALANCE_WHEATSTONE_BRIDGE:

          decodeBalanceBridgeCommand(payload, sizeOfPayload);
          break;

        default:

          transmitError(UNRECOGNIZED_COMMAND_RECEIVED);
          break;
      }

    }
}

static inline void
decodeBalanceBridgeCommand(uint8_t *payload, uint8_t sizeOfPayload)
{
    uint8_t confirm_payload[4];
    float GUISpecBalanceVolts;
    float GUISpecBalanceHz;
    void *ptr;

    if (BALANCE_PAYLOAD_SIZE != sizeOfPayload) {

      transmitError(RECEIVED_PACKET_WITH_INCORRECT_SIZE);

    } else {

      START_ATOMIC();//begin critical section; must be atomic!
      sensorRBridgeTableValid = false;

      /* floating point entries begin from startpayload[2], set ptr there */
      ptr = &payload[2];
      GUISpecBalanceVolts = byte_array_to_float(ptr);
      ptr += sizeof(float);
      GUISpecBalanceHz = byte_array_to_float(ptr);

      if (GUISpecBalanceVolts > FULLSCALE_BRIDGE_DAC_VOLTAGE || GUISpecBalanceVolts < 0.0) {

        transmitError(BRIDGE_BALANCE_VOLTAGE_OUT_OF_RANGE);
        bridge_balance_amplitude = DEFAULT_BALANCE_AMPLITUDE;

      } else {

        bridge_balance_amplitude = _Q15ftoi(GUISpecBalanceVolts / FULLSCALE_BRIDGE_DAC_VOLTAGE);

      }

      if (GUISpecBalanceHz > MAX_OUTPUT_HZ || GUISpecBalanceHz < 0) {

        transmitError(BRIDGE_BALANCE_FREQUENCY_OUT_OF_RANGE);
        bridge_balance_frequency = DEFAULT_BALANCE_FREQUENCY;
```

```
        } else {

            bridge_balance_frequency = _Q15ftoi(GUISpecBalanceHz * TWICE_SAMPLE_PERIOD);

        }

        confirm_payload[0] = CONFIRM_BALANCE_BRIDGE;
        confirm_payload[1] = 0;
        confirm_payload[2] = confirm_payload[0] ^ confirm_payload[1];
        send(confirm_payload, 3);

        START_ATOMIC();//begin critical section; must be atomic!
        global_state = RAMP_DOWN_COIL_BALANCE_BRIDGE;
        END_ATOMIC();//end critical section
    }
}

static inline void
decodeStartCommand(uint8_t startpayload[], uint8_t sizeOfPayload)
{
    uint8_t *ptr;
    float GUISpecifiedA1;
    float GUISpecifiedF1;
    float GUISpecifiedA2;
    float GUISpecifiedF2;
    float GUISpecifiedT;
    uint8_t GUISpeciedBridgeGainFactor;
    float GUISpecifiedBridgeAnalogGain = 30.0;

    if (START_PAYLOAD_SIZE != sizeOfPayload) {
        transmitError(RECEIVED_PACKET_WITH_INCORRECT_SIZE);
        return;
    }

    /* floating point entries begin from startpayload[2], set ptr there */
    ptr = &startpayload[2];
    GUISpecifiedA1 = byte_array_to_float(ptr);

    ptr += sizeof(float);
    GUISpecifiedF1 = byte_array_to_float(ptr);

    ptr += sizeof(float);
    GUISpecifiedA2 = byte_array_to_float(ptr);

    ptr += sizeof(float);
    GUISpecifiedF2 = byte_array_to_float(ptr);

    ptr += sizeof(float);
    GUISpecifiedT = byte_array_to_float(ptr);

    ptr += sizeof(float);
    GUISpecifiedBridgeAnalogGain = byte_array_to_float(ptr);

    ptr += sizeof(float);
    GUISpeciedBridgeGainFactor = *ptr;

    processStartCommand(GUISpecifiedA1, GUISpecifiedF1, GUISpecifiedA2,
                        GUISpecifiedF2, GUISpecifiedT,
                        GUISpeciedBridgeGainFactor,
                        GUISpecifiedBridgeAnalogGain);

}

static float byte_array_to_float(__eds__ uint8_t *ptr)
{
    /* warning!  This assumes the host is big-endian! */
    float myFloat;
    __eds__ uint8_t *fl_ptr;
    fl_ptr = (__eds__ uint8_t *)(&myFloat);

    *fl_ptr = *ptr;
```

```
    ++fl_ptr; ++ptr;
    *fl_ptr = *ptr;
    ++fl_ptr; ++ptr;
    *fl_ptr = *ptr;
    ++fl_ptr; ++ptr;
    *fl_ptr = *ptr;

    return myFloat;
}

static void float_to_byte_array(float input, __eds__ uint8_t *ptr)
{
    /* warning!  This assumes the host is big-endian! */
    __eds__ uint8_t *fl_ptr;
    fl_ptr = (__eds__ int8_t *)(&input);

    *ptr = *fl_ptr;
    ++ptr; ++fl_ptr;
    *ptr = *fl_ptr;
    ++ptr; ++fl_ptr;
    *ptr = *fl_ptr;
    ++ptr; ++fl_ptr;
    *ptr = *fl_ptr;
}


/*
 * utility.c
 *
 * Author: Michael Reinhart Sandstedt
 *
 * First release: September 13th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#include "spintronicsIncludes.h"
#include "spintronicsStructs.h"

inline void START_ATOMIC(void)
{
    //call this twice in case an interrupt occurs during the first call
    __asm__ volatile ("DISI #0x3FFF");
    __asm__ volatile ("DISI #0x3FFF");
}

inline void END_ATOMIC(void)
{
    DISICNT = 0;
}

inline void NOP(void)
{
    __asm__ volatile ("nop");
}
```

Header Files Content

```
/*
 * File:   AAF_STG12.h
 * Author: Michael Sandstedt
 *
 * Created on January 3, 2014, 9:52 PM
 * Last modified: 11-Jan-2014
 */

#ifndef AAF_STG12_H
#define AAF_STG12_H

#ifndef SPINTRONICS_INCLUDES_H
#include "../spintronicsIncludes.h"
#endif

   #define AAF_STG12_TAPS 128
   #define AAF_STG12_TAPS_LESS1 127
   #define AAF_STG12_TAPS_BY2_LESS2 62

   #define AAF_STG1_DLY_SZ 159 // must be AAF_STG12_TAPS + 31
   #define AAF_STG1_DLY_SZ_LESS1 158
   #define AAF_STG1_DLY_END 635 // offset to last byte in the array

   #define AAF_STG2_DLY_SZ_LESS1 127
   #define AAF_STG2_DLY_END 511 // offset to last byte in the array

   static const _Q15 AAF_STG12[128] __attribute__((space(xmemory), eds, aligned)) = {38, 38, 39, 40, 42, 44, 47, 51, 55, 59, 64, 69,
75, 82, 88, 95, 103, 111, 119, 128, 137, 146, 155, 165, 175, 185, 196, 206, 217, 227, 238, 249, 260, 271, 282, 292, 303, 314, 324, 334,
344, 354, 364, 373, 382, 391, 399, 408, 415, 423, 430, 436, 442, 448, 453, 458, 462, 465, 469, 471, 473, 475, 476, 477, 477, 476, 475,
473, 471, 469, 465, 462, 458, 453, 448, 442, 436, 430, 423, 415, 408, 399, 391, 382, 373, 364, 354, 344, 334, 324, 314, 303, 292, 282,
271, 260, 249, 238, 227, 217, 206, 196, 185, 175, 165, 155, 146, 137, 128, 119, 111, 103, 95, 88, 82, 75, 69, 64, 59, 55, 51, 47, 44, 42,
40, 39, 38, 38};

#endif       /* AAF_STG12_H */

/*
 * balanceBridge.h
 *
 * Author: Michael Reinhart Sandstedt
 *
 * First release: September 12th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#ifndef BALANCE_BRIDGE_H
#define BALANCE_BRIDGE_H

extern void balanceBridgeFSM(void);

extern uint16_t sensorRBridgeTable[256];
extern bool sensorRBridgeTableValid;

#endif


/*
 * calculateVectors.h
 *
 * Author: Michael Reinhart Sandstedt
```

```
 *
 * First release: September 14th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */
#ifndef CALCULATE_VECTORS_H
#define CALCULATE_VECTORS_H

#ifndef SPINTRONICS_STRUCTS_H
#include "spintronicsStructs.h"
#endif

extern inline void spawnVectorCalcThread(uint16_t delayCycles, uint8_t sensorIndex, uint64_t *cosAccumulator, uint64_t
*sinAccumulator, bool bridgeADCClip, bool coilADCClip, bool bridgeDigitalClip, bool f1PlusF2OutOfRange, float
implementedBridgeGain);
extern void calculateFinalVectors(void);

extern uint8_t sensorAddressCapture;
extern Q31_IQ_array_t IQ_capture;
extern bool bridgeADCClipCapture;
extern bool coilADCClipCapture;
extern bool bridgeDigitalClipCapture;
extern bool f1PlusF2OutOfRangeCapture;
extern float inverseBridgeAnalogGainCapture;

#endif


/*
 * File:   commsDefines.h
 * Author: Michael R Sandstedt
 *
 * Created on September 15, 2013, 8:16 PM
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

//transport layer flags
#define START_FLAG 0x12
#define STOP_FLAG 0x13
#define ESCAPE_FLAG 0x7D
#define ESCAPE_XOR 0x20

//message type received from GUI
#define START_COMMAND              0x00
#define STOP_COMMAND               0x01
#define CONFIG_MUX_ADDRESSING      0x04
#define BALANCE_WHEATSTONE_BRIDGE  0x05

#define MAX_MUX_ADDRESS_TABLE_SIZE 64

#define START_PAYLOAD_SIZE         28
#define STOP_PAYLOAD_SIZE          3
#define BALANCE_PAYLOAD_SIZE       11
#define MAX_RX_PAYLOAD_SIZE        67 // (MAX_MUX_ADDRESS_TABLE_SIZE + 3)

//message type sent to GUI
#define CONFIRM_START_COMMAND      0x80
#define CONFIRM_STOP_COMMAND       0x81
#define REPORT_VALUES              0x82
#define REPORT_ERROR               0x83
#define CONFIRM_MUX_ADDRESSING     0x84
#define CONFIRM_BALANCE_BRIDGE     0x85

//error codes
#define Bad_Packet_XOR 0x01
#define A1_OUT_OF_RANGE 0x02
#define F1_OUT_OF_RANGE 0x03
#define A2_OUT_OF_RANGE 0x04
```

```
#define F2_OUT_OF_RANGE 0x05
#define F1_PLUS_F2_OUT_OF_RANGE 0x06
#define T_OUT_OF_RANGE 0x07
#define INVALID_DIGITAL_GAIN_VALUE 0x08
#define BRIDGE_ADC_CLIP 0x09
#define COIL_ADC_CLIP 0x0A
#define BRIDGE_DIGITAL_CLIP 0x0B
#define ANALOG_GAIN_OUT_OF_RANGE 0x0C
#define BRIDGE_BALANCE_VOLTAGE_OUT_OF_RANGE 0x0D
#define BRIDGE_BALANCE_FREQUENCY_OUT_OF_RANGE 0x0E
#define BRIDGE_BALANCE_FAILURE 0x0F
#define RECEIVED_PACKET_WITH_INCORRECT_SIZE 0x10
#define UNRECOGNIZED_COMMAND_RECEIVED 0x11
#define UART_RX_BUFFER_OVERFLOW 0x12
#define MALFORMED_PACKET_RECEIVED 0x13


/*
 * File:   constants.h
 * Author: Michael R Sandstedt
 *
 * Created on September 16, 2013, 9:34 PM
 */

//these values are accurate
#define PROCESSOR_CYCLES_PER_MS 70000
#define CURRENT_SENSE_RESISTOR_OHMS 10e-3
#define ADC_FULLSCALE_VOLTS 2.8
#define SCALE_DIVIDER 1073741824//2^31 / 2, from sqrt(Q31 A/2 * Q31 A/2 + Q31 A/2 * Q31 A/2)

//these values depend upon the sample rate
#define SAMPLE_RATE 32000 //units are samples per second
#define HALF_SAMPLE_RATE 16000.0 //SAMPLE_RATE * 0.5
#define TWICE_SAMPLE_PERIOD 62.5e-6 //1.0 / HALF_SAMPLE_RATE
#define MAX_OUTPUT_HZ 7400.0 //input BPF cuts off at 7400, stop - band lobing starts at ~0.49 * HALF_SAMPLE_RATE
#define MAX_OUTPUT_FREQUENCY 16056//_Q15ftoi(MAX_OUTPUT_HZ * twice_sample_period)
#define DEFAULT_BALANCE_FREQUENCY 2048//1000Hz, _Q15ftoi(1000.0 * twice_sample_period);
#define DELAY_TO_60HZ_PI 267//SAMPLE_RATE / (60Hz * 2)

//TODO: these values need calibration
#define FULLSCALE_BRIDGE_DAC_VOLTAGE 2.8297521 //units are volts //the Wheatstone bridge voltage that corresponds to a full-scale output as of 12/22/2013 // the DAC outputs 1.25Vpeak on two differential pins giving 2.5peak total; the DAC buffer gain is 6.04k/4.99k; 2.5 * 6.04 / 4.99 = 3.02605210
#define FULLSCALE_COIL_DAC_VOLTAGE .763//untis are volts//the output voltage measured on Nov 30th, 2013 at the coil amplifier output corresponding to a full-scale output at 50Hz//TODO: should we be measuring pre-amplifier, or should we perhaps measure current instead?
#define U2_BUF_GAIN 1.39819005
#define U2_INVERSE_GAIN .715210354
#define COIL_ADC_BUFFER_GAIN 20.179

//TODO: these values depend upon the above calibrated values
#define DEFAULT_BALANCE_AMPLITUDE 14834 //1Volt, _Q15ftoi(1.0 / FULLSCALE_BRIDGE_DAC_VOLTAGE);
#define COIL_ADC_SCALE_FACTOR 12.9228565e-9 //ADC_FULLSCALE_VOLTS / CURRENT_SENSE_RESISTOR_OHMS / COIL_ADC_BUFFER_GAIN / SCALE_DIVIDER
#define BRIDGE_ADC_SCALE_FACTOR 2.60770321e-9 //ADC_FULLSCALE_VOLTS / SCALE_DIVIDER
#define BRIDGE_ADC_SCALE_FACTOR_BY_2 1.3038516e-9 //BRIDGE_ADC_SCALE_FACTOR / 2.0
#define BRIDGE_ADC_SCALE_FACTOR_BY_4 651.925802e-12 //BRIDGE_ADC_SCALE_FACTOR / 4.0
#define BRIDGE_ADC_SCALE_FACTOR_BY_8 325.962901e-12 //BRIDGE_ADC_SCALE_FACTOR / 8.0
#define BRIDGE_ADC_SCALE_FACTOR_BY_16 162.981451e-12 //BRIDGE_ADC_SCALE_FACTOR / 16.0
#define BRIDGE_ADC_BUFFER_MIN_GAIN 9.38784748 //getBridgeBufGainFromU24Code(0)
#define BRIDGE_ADC_BUFFER_MAX_GAIN 157.022822 //getBridgeBufGainFromU24Code(0xFF)
#define INVERSE_BRIDGE_ANALOG_MIN_GAIN 106.520691e-3
#define INVERSE_BRIDGE_ANALOG_MAX_GAIN 6.36850101e-3

#define FIXED_ANALOG_GAIN_VALUE 32.4690801

/*
 * CS4272.h
 *
 * designed and written
 * by Michael Reinhart Sandstedt
```

```
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

extern void cs4272Init(void);

/*
 * digiPotDrv.h
 *
 * Author: Michael Reinhart Sandstedt
 *
 * First release: September 11th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#ifndef DIGI_POT_DRV_H
#define DIGI_POT_DRV_H

//see getRBridgeOhms(uint16_t val) for corresponding values in ohms
#define R_BRIDGE_MAX 1535
#define R_BRIDGE_MIN 0
#define R_BRIDGE_MID 768

//see getRAmpOhms(uint8_t val) for corresponding values in ohms
#define R_AMP_MAX 255
#define R_AMP_MIN 0
#define R_AMP_MID 128

extern void setRBridge(uint16_t val);
extern void setRAmp(uint8_t val);
extern inline float getBridgeBufGainFromU24Code(uint8_t u24_code);
extern inline float getBridgeInverseGainFromU24Code(uint8_t u24_code);
extern inline uint8_t getU24CodeFromBrdigeBufGain(float bridge_gain);

#endif

/*
 * File:   FIR.h
 * Author: Michael R Sandstedt
 *
 * Created on December 2, 2013, 10:52 PM
 */

#ifndef FIR_H
#define      FIR_H

#ifndef SPINTRONICS_INCLUDES_H
#include "spintronicsIncludes.h"
#endif

#define MASK_START_NOT_FSUM 0x4000
#define MASK_START_EVERYTHING 0xC000
#define MAST_START_FSUM 0x8000
#define MASK_T 0x3FFF

extern void firInit(void);
extern inline int16_t FIR(int16_t input);
extern inline bool lpf_decimate(uint16_t T_by_1024, _Q31_t stg3_coef,
       Q31_IQ_array_t input, __eds__ Q31_IQ_array_t *results);

#endif       /* FIR_H */


/*
 * File:   fsmStates.h
 * Author: Michael R Sandstedt
```

```
 *
 * Created on September 15, 2013, 8:11 PM
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

//states for bridgeBalanceFSM() must have bit 7 set!
#define BALANCE_BRIDGE_FSM_MASK             0x40
#define IDLE                                0x40
#define START_BRIDGE_BALANCE_FSM            0x41
#define START_GAIN_CAL                      0x42
#define SET_R_AMP_TO_LO_MID                 0x43
#define R_AMP_LO_MID_SIGNAL_SETTLING        0x44
#define R_AMP_LO_MID_CLIP_TEST              0x45
#define SET_R_AMP_TO_HI_MID                 0x46
#define R_AMP_HI_MID_SIGNAL_SETTLING        0x47
#define R_AMP_HI_MID_CLIP_TEST              0x48
#define START_BRIDGE_CAL                    0x49
#define SET_R_BRIDGE_TO_LO_MID              0x4A
#define R_BRIDGE_LO_MID_SIGNAL_SETTLING     0x4B
#define R_BRIDGE_LO_MID_AMP_MEASURE         0x4C
#define R_BRIDGE_LO_MID_AMP_CALC            0x4D
#define SET_R_BRIDGE_TO_HI_MID              0x4E
#define R_BRIDGE_HI_MID_SIGNAL_SETTLING     0x4F
#define R_BRIDGE_HI_MID_AMP_MEASURE         0x50
#define R_BRIDGE_HI_MID_AMP_CALC            0x51

/*
 * or the state with this in order for balanceBridgeFSM() to continue to
 * measurementFSM() after the balance routine completes
 */
#define START_MEASUREMENT_AFTER_BALANCE_MASK    0x20


//states for measurementFSM() must have bit 8 set!
#define MEASUREMENT_FSM_MASK                0x80
#define START_MEASUREMENT_FSM               0x81
#define RAMP_UP_COIL                        0x82
#define START_NEW_MEASUREMENT_CYCLE         0x83
#define START_SIGNAL_GEN                    0x84
#define WAIT_FOR_COIL_0RAD                  0x85
#define     MEASURE_NOT_FSUM                0x86
#define     MEASURE                         0x87
#define     CALCULATE_VECTORS               0x88
#define RAMP_DOWN_COIL_QUIT                 0x89
#define RAMP_DOWN_COIL_RESTART              0x8A
#define RAMP_DOWN_COIL_BALANCE_BRIDGE       0x8B

//signal generator state encoding
#define RESET_SIGNAL_GEN 0
#define RESET_BRIDGE_GEN 1
#define RUN_SIGNAL_GEN 2

//UART RX FSM states
#define RX_IDLE 0x00
#define RX_MID_PAYLOAD 0x01
#define RX_ESCAPE 0x02


/*
 * generateAndProcessSamples.h
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */
```

```
extern void measurementFSM(void);


/*
 * i2sDrv.h
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

extern void i2sInit(void);

/* WARNING: this file is automatically generated by h/AAF/build_input_BPF.m; make all modifications there! */

/*
 * File:   input_BPF.h
 * Author: Michael Sandstedt
 *
 * Created on January 3, 2014, 9:52 PM
 * Last modified: 05-Jan-2014
 */

#ifndef INPUT_BPF_H
#define      INPUT_BPF_H

#ifndef SPINTRONICS_INCLUDES_H
#include "../spintronicsIncludes.h"
#endif

   #define BPF_IS_EVEN_ORDER
   #define DO_CONST "#62"
   #define INPUT_BPF_TAPS 129

   static const int16_t input_BPF[129] __attribute__((space(ymemory), eds, aligned)) = {-4, 1, 18, 15, -5, -7, 14, 21, -3, -20, 3, 28, 2, -
38, -22, 26, 13, -56, -66, 5, 24, -67, -127, -47, 27, -64, -196, -140, 5, -45, -253, -272, -63, -18, -277, -427, -197, 0, -249, -573, -403, -
22, -158, -667, -677, -125, -9, -657, -993, -354, 175, -477, -1313, -776, 363, -18, -1590, -1566, 517, 1114, -1778, -3857, 603, 9747,
14553, 9747, 603, -3857, -1778, 1114, 517, -1566, -1590, -18, 363, -776, -1313, -477, 175, -354, -993, -657, -9, -125, -677, -667, -
158, -22, -403, -573, -249, 0, -197, -427, -277, -18, -63, -272, -253, -45, 5, -140, -196, -64, 27, -47, -127, -67, 24, 5, -66, -56, 13, 26, -
22, -38, 2, 28, 3, -20, -3, 21, 14, -7, -5, 15, 18, 1, -4};

#endif       /* INPUT_BPF_H */


/*
 * muxControl.h
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

extern void muxInit (void);
extern void configSensor(uint8_t sensor);


/*
 * spintronicsConfig.h (formerly spintronics.h)
 *
 * written
 * by Michael Reinhart Sandstedt
 *
```

```
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#ifndef SPINTRONICS_CONFIG_H
#define	SPINTRONICS_CONFIG_H

#define FIXED_ANALOG_GAIN

//#define MEASURE_F2_AT_BRIDGE
//#define MEASURE_F2_AT_COIL
//#define SIMULATION_MODE //un-comment this for a simulation specific build
//#define NO_GUI //un-comment to launch a start command automatically form main
//#define CODEC_USES_I2S //un-comment this if tx/rx samples from the CODEC are in I2S format; otherwise samples are left-justified

#ifdef SIMULATION_MODE
#define MEASUREMENT_SETUP_TIME 10
#define GAIN_CHECK_SETUP_TIME 10
#define BRIDGE_CHECK_SETUP_TIME 10
#define GAIN_CHECK_MEASURE_CYCLES 1
#define BRIDGE_CHECK_MEASURE_TIME_MULTIPLIER 1
#else
//units are samples//the amount of time to let the transients settle after switching sensors
#define MEASUREMENT_SETUP_TIME 1000
#define GAIN_CHECK_SETUP_TIME 50
#define BRIDGE_CHECK_SETUP_TIME 100
#define GAIN_CHECK_MEASURE_CYCLES 2
#define BRIDGE_CHECK_MEASURE_TIME_MULTIPLIER 3
#endif

#define MAX_MEASUREMENT_SAMPLES 0x3FFFC00//max iterations for stage 3 is max(uint16_t) = 0xFFFF. stg3 operates at 1024/(sample clock); 2^10 * 0xFFFF = 0x3FFFC00
#define MIN_MEASUREMENT_SAMPLES 1024//TODO: what's our minimum measurement period before UART transmission overhead causes the state machine to fail? appropriate value TBD

#define DELAY_TO_VECTOR_CALC_THREAD 150//processor cycles

//uncomment PROBE and one of the probe sites to output that signal to the COIL DAC
//#define PROBE
//#define PROBE_POST_FIR
//#define PROBE_BRIDGE_ADC
//#define PROBE_COIL_ADC
//#define PROBE_BRIDGE_X_COS_F1_T
//#define PROBE_BRIDGE_X_COS_F2_T
//#define PROBE_BRIDGE_X_COS_FSUM_T
//#define PROBE_BRIDGE_X_COS_FDIFF_T
//#define PROBE_COIL_X_COS_F2_T
//#define PROBE_BRIDGE_X_SIN_F1_T
//#define PROBE_BRIDGE_X_SIN_F2_T
//#define PROBE_BRIDGE_X_SIN_FSUM_T
//#define PROBE_BRIDGE_X_SIN_FDIFF_T
//#define PROBE_COIL_X_SIN_F2_T
//#define PROBE_BRIDGE_X_COS_F1_T_PLUS_PHI
//#define PROBE_BRIDGE_X_COS_F2_T_PLUS_PHI
//#define PROBE_BRIDGE_X_COS_FSUM_T_PLUS_PHI
//#define PROBE_BRIDGE_X_COS_FDIFF_T_PLUS_PHI
//#define PROBE_COIL_X_COS_F2_T_PLUS_PHI

#endif


/*
 * File:   spintronicsIncludes.h
 * Author: Michael R Sandstedt
 *
 * Created on September 15, 2013, 8:08 PM
 *
 * questions or support:
```

```
 * michael.sandstedt@gmail.com
 */
#ifndef SPINTRONICS_INCLUDES_H
#define      SPINTRONICS_INCLUDES_H

#include <p33EP512GP806.h>
#include <math.h>
#include <libq.h>
#include <stdint.h>
#include <stdbool.h>
#include <stdlib.h>

#endif


/*
 * File:   spintronics_structs.h
 * Author: Michael R Sandstedt
 *
 * Created on November 28, 2013, 11:29 AM
 */

#ifndef SPINTRONICS_STRUCTS_H
#define      SPINTRONICS_STRUCTS_H

#ifndef SPINTRONICS_CONFIG_H
#include "spintronicsConfig.h"
#endif

#ifndef SPINTRONICS_INCLUDES_H
#include "spintronicsIncludes.h"
#endif

typedef struct accumulator_s {
   int64_t bridge_f1;
#ifdef MEASURE_F2_AT_BRIDGE
   int64_t bridge_f2;
#endif
   int64_t bridge_fdiff;
   int64_t bridge_fsum;
#ifdef MEASURE_F2_AT_COIL
   int64_t coil_f2;
#endif
} accumulator_t;

typedef struct float_array_s {
   float bridge_f1;
#ifdef MEASURE_F2_AT_BRIDGE
   float bridge_f2;
#endif
   float bridge_fdiff;
   float bridge_fsum;
#ifdef MEASURE_F2_AT_COIL
   float coil_f2;
#endif
} float_array_t;

typedef struct double_array_s {
   double bridge_f1;
#ifdef MEASURE_F2_AT_BRIDGE
   double bridge_f2;
#endif
   double bridge_fdiff;
   double bridge_fsum;
#ifdef MEASURE_F2_AT_COIL
   double coil_f2;
#endif
} double_array_t;

typedef struct angle_array_s {
   _Q15 two_f1_t;
```

```
    _Q15 two_f2_t;
    _Q15 two_fdiff_t;
    _Q15 two_fsum_t;
} angle_array_t;

typedef struct _Q23_aligned_s {
    uint8_t wordAlign;
    uint8_t lowByte;
    uint16_t Q15;
} _Q23_aligned_t;

typedef struct dword_s {
    uint16_t lowWord;
    int16_t highWord;
} dword_t;

typedef struct qbyte_s {
    uint8_t byte0;
    uint8_t byte1;
    uint8_t byte2;
    int8_t byte3;
} qbyte_t;

typedef union _Q31_u {
    qbyte_t qbyte;
    dword_t dword;
    _Q23_aligned_t Q23_aligned;
    int32_t Q31;
} _Q31_t;

typedef struct Q31_IQ_array_s {
    _Q31_t f1_I;
    _Q31_t f1_Q;
#ifdef MEASURE_F2_AT_BRIDGE
    _Q31_t bridge_f2_I;
    _Q31_t bridge_f2_Q;
#endif
    _Q31_t fdiff_I;
    _Q31_t fdiff_Q;
    _Q31_t fsum_I;
    _Q31_t fsum_Q;
#ifdef MEASURE_F2_AT_COIL
    _Q31_t coil_f2_I;
    _Q31_t coil_f2_Q;
#endif
} Q31_IQ_array_t;

typedef union float_bytes_u {
    qbyte_t qbyte;
    uint32_t u32;
    float fl;
}float_bytes_t;

#endif      /* SPINTRONICS_STRUCTS_H */



/*
 * spiTx.h
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 *
 * First release: September 10th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

//addresses for U20, U23 and U24 to be used for spiTx();
#define U20 0x10
```

```
#define U23 0x20
#define U24 0x30

extern void spiInit(void);
extern void spiTx(uint8_t addr, uint8_t pl);


/*
 * File:   timerDrv.h
 * Author: Michael R Sandstedt
 *
 * Created on September 15, 2013, 8:52 PM
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

extern void timerInit(void);
extern void busy_wait_ms(uint16_t ms);


/*
 * uartDrv.h
 *
 * designed and written
 * by Michael Reinhart Sandstedt
 *
 * First release: May 7th, 2013
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#ifndef SPINTRONICS_STRUCTS_H
#include "spintronicsStructs.h"
#endif

#ifndef SPINTRONICS_CONFIG_H
#include "spintronicsConfig.h"
#endif

//function prototypes
extern void uartInit (void);
extern inline void transmitResults(uint8_t sensor,
                            __eds__ float_array_t *phaseAngle,
                            __eds__ float_array_t *amplitude,
                            bool bridgeADCClip, bool coilADCClip,
                            bool bridgeDigitalClip);
extern inline void transmitError(uint8_t errorCode);
#if defined(NO_GUI) || defined(SIMULATION_MODE)
inline void processStartCommand(float GUISpecifiedA1, float GUISpecifiedF1,
                            float GUISpecifiedA2, float GUISpecifiedF2,
                            float GUISpecifiedT,
                            uint8_t GUISpeciedBridgeGainFactor,
                            float GUISpecifiedBridgeAnalogGain);
#endif

//global variables
extern uint8_t global_state;
extern uint16_t measurementTime;//units are samples / 1024
extern _Q31_t stg3_coef;//
extern _Q15 f1;//units are half-cycles per sample-period
extern _Q15 f2;//units are half-cycles per sample-period
extern _Q15 fdiff;//units are half-cycles per sample-period
extern _Q15 fsum;//units are half-cycles per sample-period
extern _Q15 a1;
extern _Q15 a2;
extern uint8_t *sensorAddressTable;
extern uint8_t numberOfSensors;
extern bool f1PlusF2OutOfRange;
extern _Q15 bridge_balance_amplitude;
```

```
extern _Q15 bridge_balance_frequency;
extern uint8_t latest_command;


/*****************************************************************
 * The gain factor can be 0 to 8;
 * Gain = 2^bridgeADCGainFactor;
 *
 * All internal processing is 16 bit.  Thus, the lowest 8 bits of
 * incoming 24bit ADC samples are truncated. This parameter left shifts
 * the 24bit samples from the Wheatstone bridge ADC prior to truncation
 *
 * Note that the communication protocol specifies Gain explicitly
 * (e.g. 1, 2, 4, 8, 16).  Also, the protocol does not allow for
 * values above 16.  BUT, digital gain stage can actually deal with
 * all possible values: 1, 2, 4, 8, 16, 32, 64, 128, and 256
 * To extend this functionality, add cases to this switch statement:
 * switch(GUISpeciedBridgeGainFactor) in uartDrv.c/processStartCommand()
 * and to this switch statement:
 * switch(bridgeADCGainFactor) in uartDrv.c/transmitResults
*****************************************************************/
extern uint8_t bridgeADCGainFactor;
extern uint8_t u24_code;
extern float inverseBridgeAnalogGain;


/*
 * File:   utility.h
 * Author: Michael R Sandstedt
 *
 * Created on September 15, 2013, 8:21 PM
 *
 * questions or support:
 * michael.sandstedt@gmail.com
 */

#ifndef UTILITY_H
#define UTILITY_H

#ifndef SPINTRONICS_INCLUDES_H
#include "spintronicsIncludes.h"
#endif

#ifndef SPINTRONICS_STRUCTS_H
#include "spintronicsStructs.h"
#endif

int32_t asm16X16Mult(_Q15, _Q15);
extern inline void START_ATOMIC(void);
extern inline void END_ATOMIC(void);
extern inline void NOP(void);

#endif
```