Voice Control with Google Assistant

Voice Control with Google Assistant

Introduction

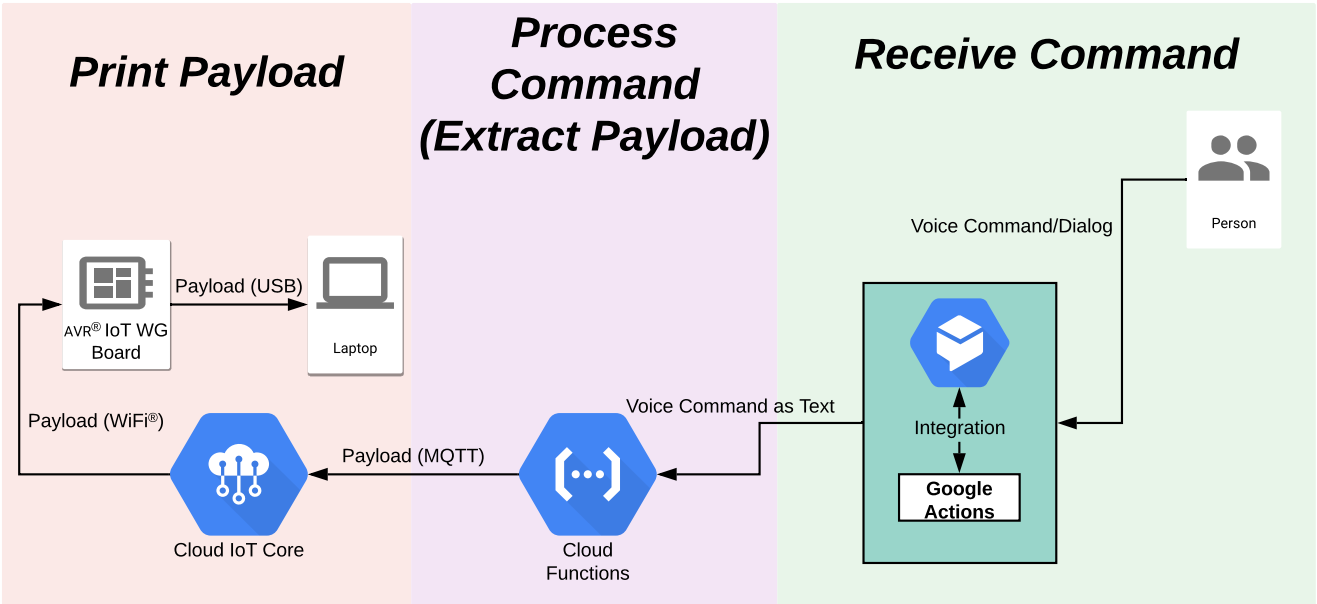
Author: Johan Lofstad, Microchip Technology Inc.

This user guide describes how the Google Home Assistant can be integrated into IoT projects using the Google Cloud Platform (GCP). Specifically, a "*print number*" example using the AVR®-IoT WG Board to print a number. When a voice command is given, such as "print 123 to the console", the IoT Board prints the number to the command line. The topics that are covered are:

- Connecting the AVR-IoT WG Board to the Cloud
- Integrating a Google Home Assistant-enabled device to a Google Cloud Project using *Dialogflow* and *Google Actions*
- Adding a voice command to send a "print x" message to the AVR-IoT Board

Figure 1 shows a flowchart of the print message example. A person initiates a voice command by speaking to a Google Home Assistant-enabled device. The voice is processed through Dialogflow and Google Actions, figuring out the command from the voice. The command is then forwarded to a *Cloud Function* as text. The cloud function processes the request and forwards the message to the AVR-IoT Board using the *Cloud IoT Core*. When the board receives the message, it is promptly sent to a connected laptop through USB.

Figure 1. Flowchart of the Print Message Example



Tip:
The example source code for the IoT Board can be found on Atmel START: https://start.atmel.com/#example/Atmel%3AAVR_IoT_WG_Sensor_Node_With_Voice_Control%3A1.0.0%3A%3AApplication%3AAVR_IoT_WG_Sensor_Node_With_Voice_Control%3A

The example source code for the Google Cloud Platform can be found on GitHub: https://github.com/microchip-pic-avr-solutions/avr-iot-wg-board-voice-control-cloud

Table of Contents

1. Cloud Configuration

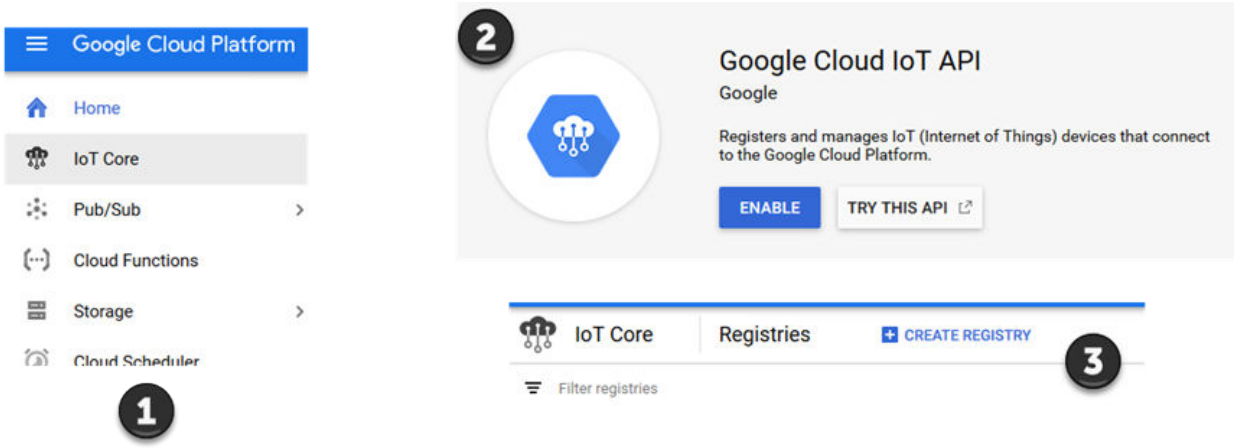
A Google Cloud Platform account and project are required to integrate a Google Assistant device. Navigate to https://cloud.google.com and create an account. When prompted to create a project, name it something relevant such as *Voice to AVR*. It might take several minutes for the project to become active.

1.1 Configuring the IoT Core

The IoT Core module is designed to handle all communication with IoT devices. All devices are registered in the IoT Core with a unique ID and authentication credentials. In short, it serves as a *gateway* between the IoT devices and the rest of the cloud.

The IoT Core module can be opened through the menu at the left-hand side of the cloud console. Click “Enable” to add the module to the cloud project. When the IoT Core is added, a *Registries* page should appear. See Figure 1-1 for screenshots of the procedure.

Figure 1-1. How to Find the IoT Core Module



The device must be added to a registry to connect to the IoT Core Module. A registry is a set of devices that can communicate with the cloud. To create a new registry, click the **Create Registry** button. There are several required fields. Configure the registry according to the “Entry” column in Table 1-1. Some fields might not appear before clicking “Show Advanced Options”.

Table 1-1. IoT Core Create Registry Fields

Name	Entry	Description
Registry ID	voice-devices	A permanent ID which identifies the registry
Region	The region which is applicable	The geographical region where the data are stored
Protocol	✓ MQTT □ HTTP	The communications protocol which the registry supports. Both MQTT and HTTP are supported.
Cloud Pub/Sub topics	Select the dropdown menu and select **Create a topic**. Enter the topic name *voices-upstream*. Leave the rest as default and press “Create topic”.	The default telemetry topic is the MQTT topic, in which all messages from the device are routed to.

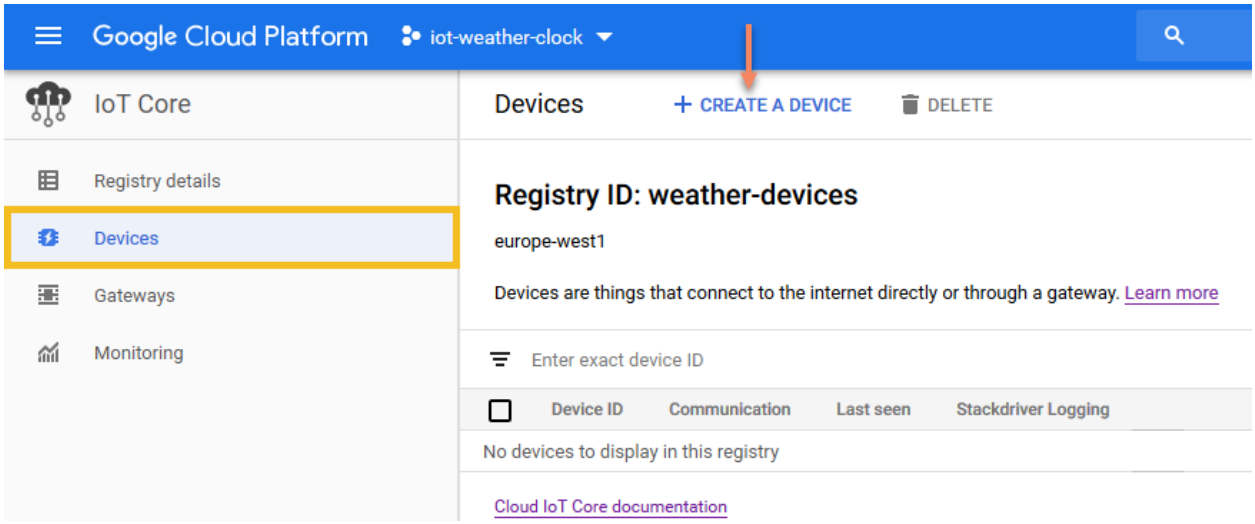
.....continued		
Name	**Entry**	**Description**
Device state topic (optional)	Leave unchanged	All state events published by the device is sent there. Not used in this example.
Stackdriver Logging	None	Not used in this example

1.1.1 Adding a Device to the Registry

All devices in a registry are found by selecting the *devices* tab on the left-hand side. A new device can be added by pressing **Create a Device**. See Figure 1-3. To add the AVR-IoT WG board, leave everything as default except "Device ID", "Public Key Format" and "Public key value". The *Device ID* is found in the URL of the "CLICK-ME.htm" file. The "CLICK-ME.html" file is located under the "CURIOSITY" drive when the kit is connected through USB, see the example in Figure 1-2. **Google Cloud requires the first character to be a letter. The entered Device ID should thus be "d + the number". For instance "d0123710B94CEB0ECFE".**

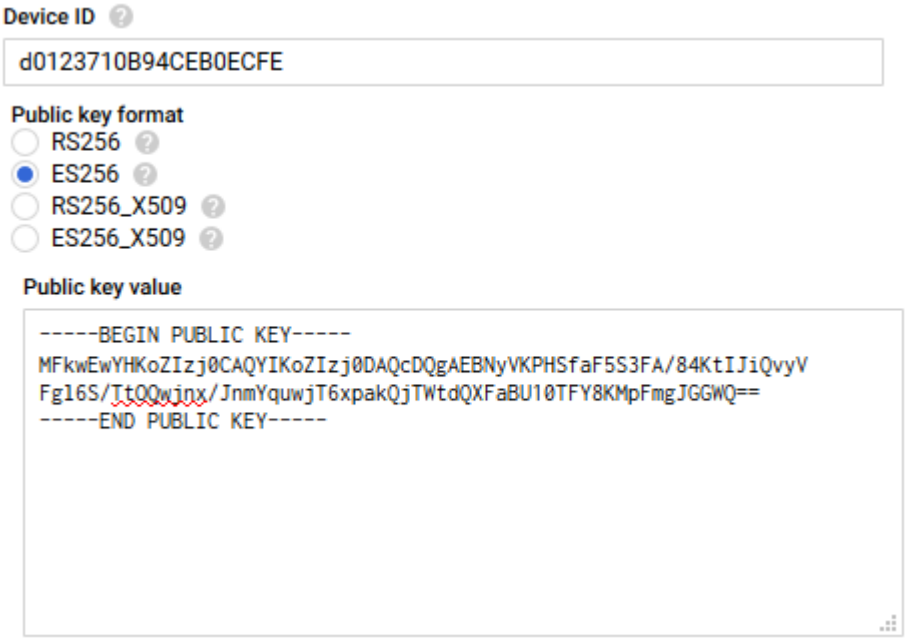
Figure 1-2. Finding the Device ID for the IoT Board

Figure 1-3. Adding a New Device to the IoT Core Registry



The public key format is "ES256". The public key is found in the "PUBKEY.txt" file under the CURIOSITY drive. Copy the contents in the public key value field. The details should be similar to Figure 1-4. Click "Create" to add the device.

Figure 1-4. Device Settings for the AVR-IoT WG Board



1.2 Setting up a Cloud Function

A *Cloud Function* is a code snippet that runs whenever a *trigger* event occurs. A cloud function can be used to receive voice command requests, process them, and forward the processed messages to the device. To create a new cloud function, navigate to the Cloud Functions module by opening it through the menu on the left-hand side of the cloud console, followed by “create function”. See Figure 1-5 for screenshots on setting up the Cloud Function.

Fill in the fields as per Table 1-2.

Figure 1-5. Creating a Cloud Function

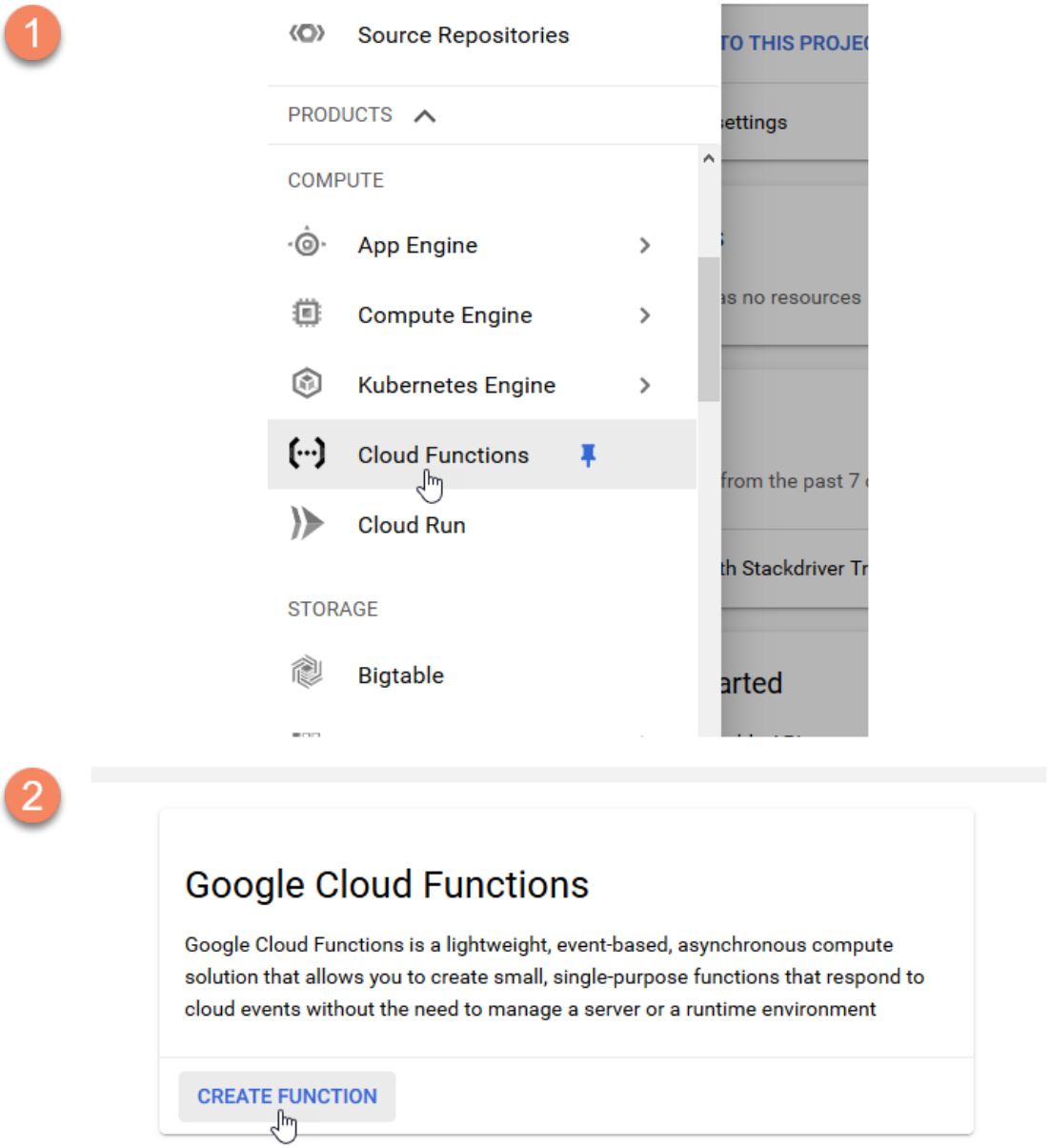


Table 1-2. Cloud Function Fields

Name	Entry	Description
Name	voice-command-process	The name of the cloud function
Trigger	HTTP	The method of triggering the cloud function
Source Code	Inline editor	The source code that is being executed. By using "*Inline editor*", the source code can be entered in the field below.

.....continued

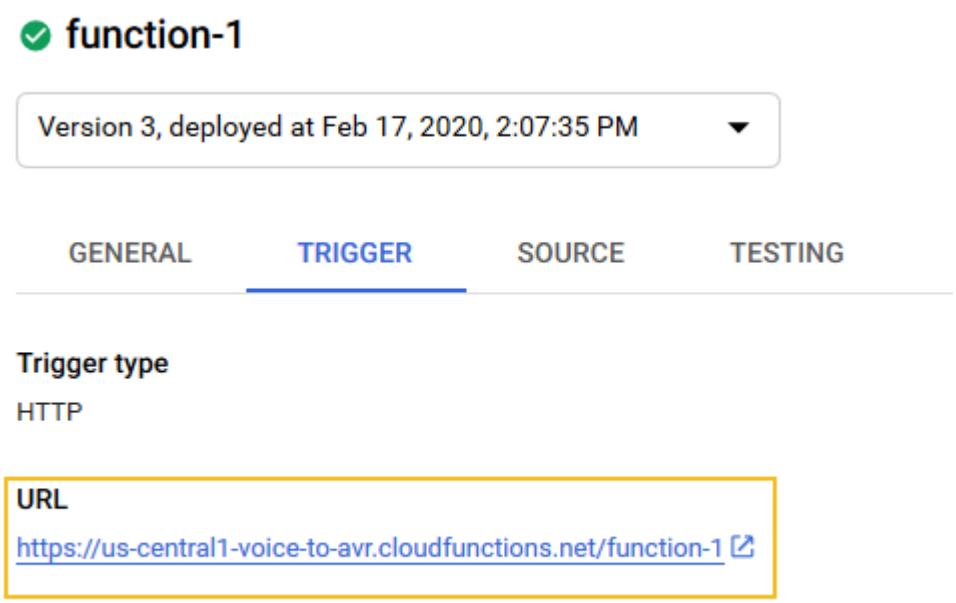
Name	Entry	Description
Runtime	Python 3.7	The language of the source code
main.py	See code below	The source code to be executed when the cloud function is triggered

requirements.txt	`google-cloud-storage`
	`google-auth==1.6.2`
	`google-api-python-client==1.7.8`
	`google-auth-httplib2==0.0.3`
	`google-cloud-pubsub==0.39.1`
	`paho-mqtt==1.4.0`
	`pyjwt==1.7.1`
	`oauth2client`
		The required python modules to run the source code

Function to execute	process_voice	The entry point of the source code

When the cloud function has been created, open it, and select the *Trigger* tab. The cloud function is invoked whenever a call is made to the listed URL. See Figure 1-6 for a screenshot. Take note of the trigger URL, as it will be used in the next step when configuring Dialogflow.

Figure 1-6. Where to Find the Trigger URL of a Cloud Function



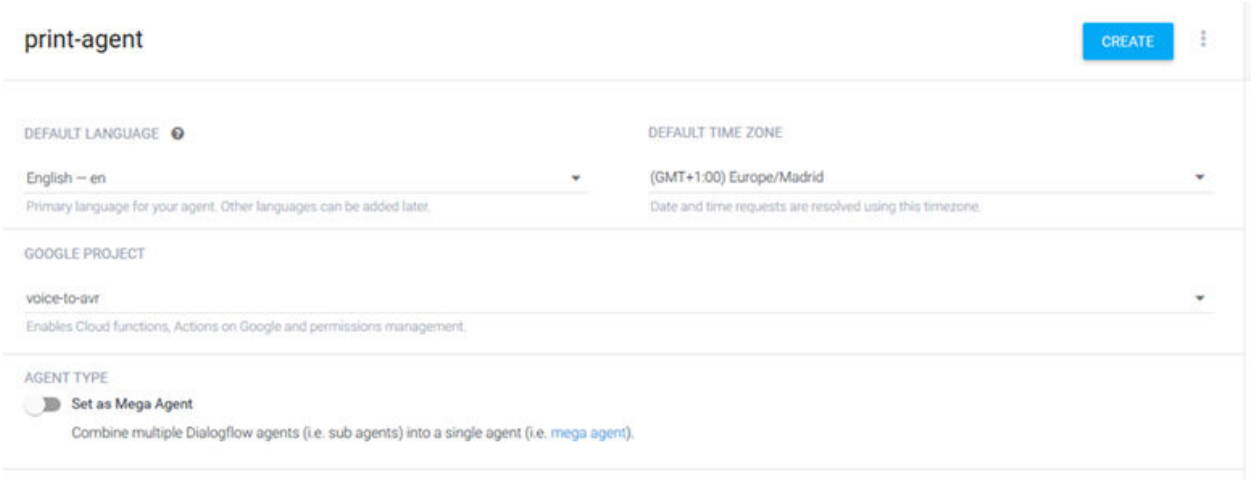

```
from googleapiclient import discovery
import base64


# IMPORTANT: Change these fields to your projects settings
PROJECT_ID = "voice-to-avr"
IOT_CORE_REGION = "europe-west1"
IOT_CORE_REGISTRY_ID = "voice-devices"
IOT_CORE_DEVICE_ID = "d0123DFDAEF65AF85FE"

# Code obtained at https://cloud.google.com
def get_gcloud_client():
    api_version = 'v1'
```

```
    discovery_api = 'https://cloudiot.googleapis.com/$discovery/rest'
    service_name = 'cloudiotcore'

    discovery_url = '{}?version={}'.format(
        discovery_api, api_version)

    return discovery.build(
        service_name,
        api_version,
        discoveryServiceUrl=discovery_url,
        credentials=None,
        cache_discovery=False)

# Code obtained at https://cloud.google.com
def send_message_to_device(project_id, cloud_region, registry_id, device_id, payload):
    """
    Sends a message to an IoT Device through the config pubsub topic. (Config pubsub is /
devices/d_id/config)
    :param project_id: Google Cloud project ID
    :param cloud_region: sWhich region is the device located in. For instance us-central1
    :param registry_id: IoT Core Registry the device is loacted in
    :param device_id: The device ID
    :param payload:
    :return:
    """
    client = get_gcloud_client()
    device_path = 'projects/{}/locations/{}/registries/{}/devices/{}'.format(
        project_id, cloud_region, registry_id, device_id)

    config_body = {
        'binaryData': base64.urlsafe_b64encode(
            payload.encode('utf-8')).decode('ascii')
    }

    return client.projects(
    ).locations().registries(
    ).devices().modifyCloudToDeviceConfig(
        name=device_path, body=config_body).execute()

def process_voice(request):
    request_json = request.get_json()
    queryResult = request_json['queryResult']
    parameters = queryResult['parameters']

    number = str(int(parameters['number']))
    payload = '{{"number":"{}"}}'.format(number)
    print("Sent {} to device".format(number))
    send_message_to_device(PROJECT_ID, IOT_CORE_REGION, IOT_CORE_REGISTRY_ID,
IOT_CORE_DEVICE_ID, payload)
```

1.3 Dialogflow and Google Actions

Dialogflow is a service that enables the integration of natural language processing and voice control into Google Cloud Applications without having to write any lines of code. For the print example, Dialogflow should call the trigger URL for the Cloud Function created in Section 1.2 Setting up a Cloud Function, with the desired number to be printed. Dialogflow can be found at https://dialogflow.cloud.google.com. When registered, a prompt to create a new agent appears. Name it “print-agent”, select English as the default language and the Google project to the one created in previous sections. Click *Create* to finalize the agent. See Figure 1-7 for a screenshot of the creation process.

Figure 1-7. Creating a Dialogflow Agent



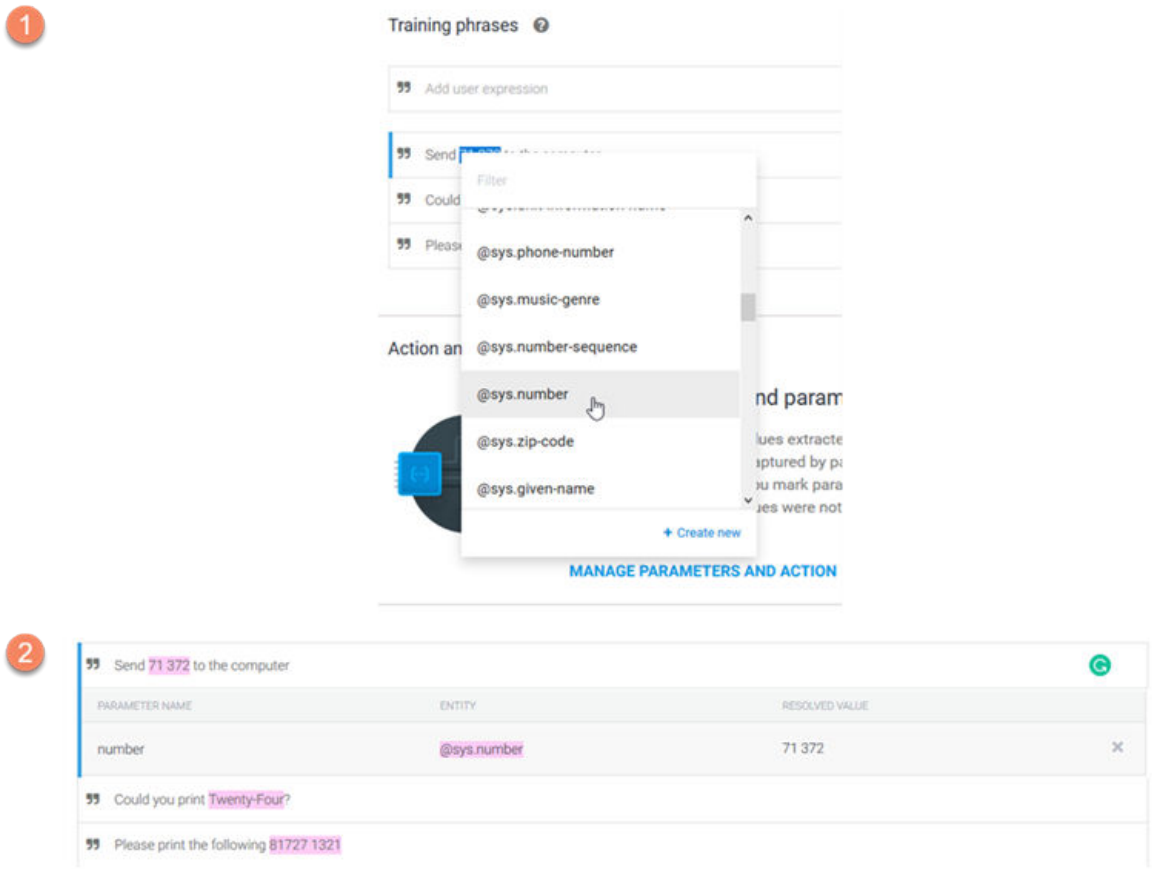
1.3.1 Creating an Intent

Dialogflow works with the concept of *Intents*. An *Intent* is something the user wants to do, something he/she *intends*. For this example, the intent is to print a given message from the IoT Board to the PC. To create an intent, click on *Create Intent*. The first step is to add some training phrases, which are examples in natural language of how the user would communicate their intent. Add the following training phrases (or more):

- Please print the following 81727 1321
- Could you print Twenty Four?
- Send 71 372 to the computer

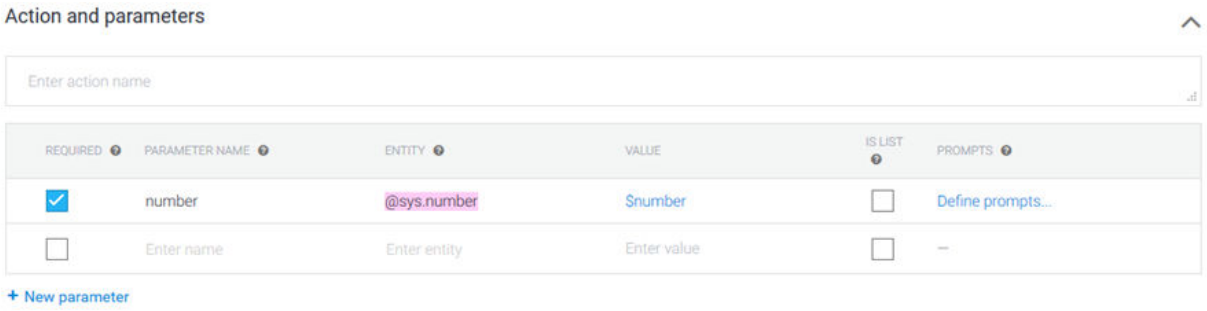
For every training phrase added, the *entity* in the phrase must be marked. An entity is a way to define what information to be extracted from the message. For this example, mark the numbers in the phrase and select *@sys.number*, identifying it as a number in the phrase. See Figure 1-8 for a screenshot of the procedure.

Figure 1-8. Adding Training Phrases to the Intent



The field *Action and Parameters* is used to define which parameter the extracted entity should be assigned to. Fill in a parameter, as shown in Figure 1-9.

Figure 1-9. Number Parameter Definition



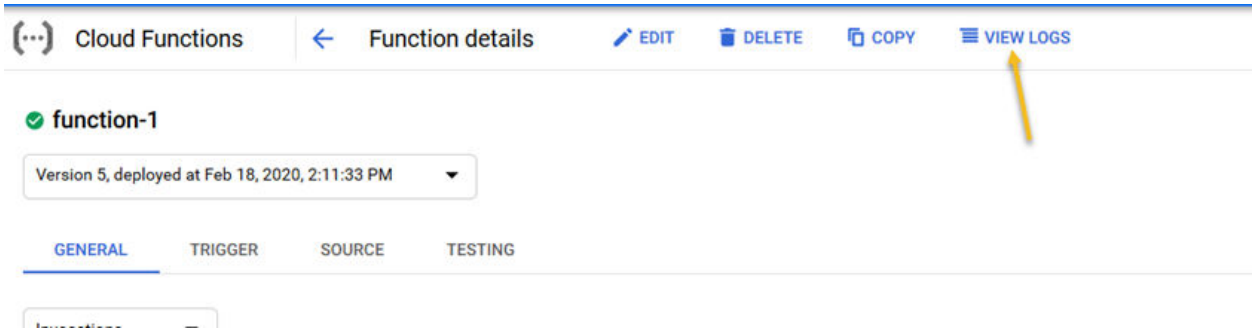
To add some flavor to the dialog, a response can be added. This response is spoken back to the user when the intent has been finished. Click on *Add Response*, and type in: *"Printing $number from the AVR-IoT Board"*. Note the "$number" part, which refers to the parameter defined above. To close the conversation after the response, check *Set this intent as end of conversation*. Finally, under *Fulfillment*, enable the slider for *Enable webhook call for this intent*. Finish by clicking *Save* in the upper right-hand corner.

A *fulfillment* is what the dialog does at the end of the conversation with the newly acquired information. Click on *Fulfillment* on the left-hand side menu and use the slider on the right-hand side to enable the webhook. In the URL field, enter the trigger URL from the cloud function created in the previous section. Leave everything else as default and click *"Save"*.

1.3.2 Testing and Verifying the Dialog

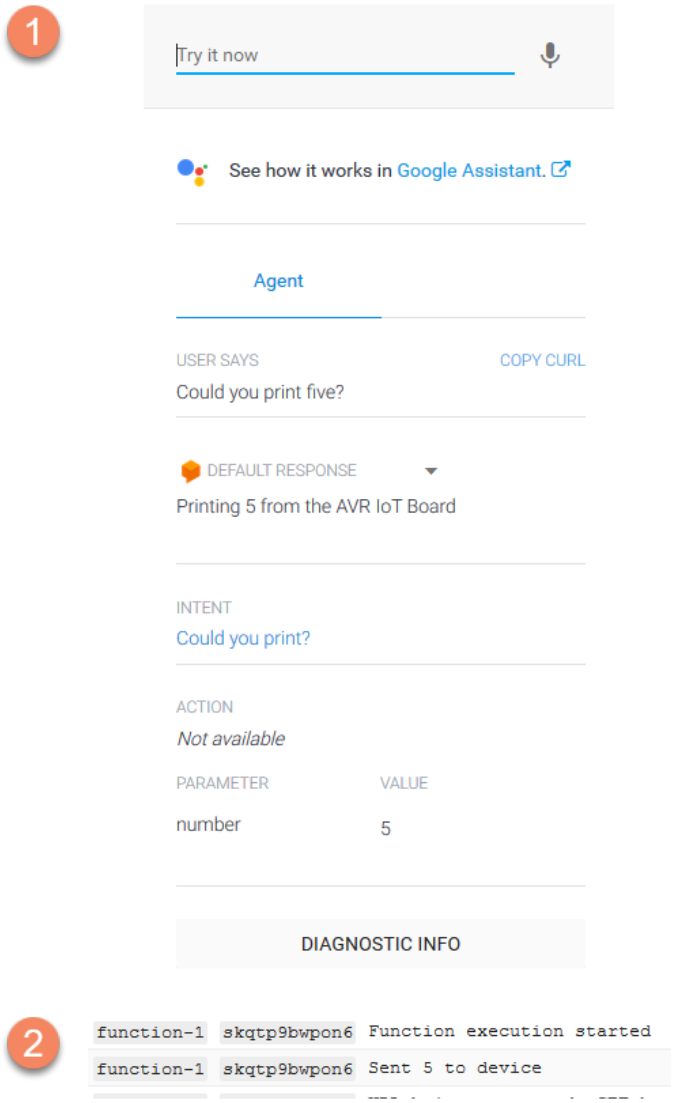
Before proceeding by adding a home assistant device and configuring the AVR-IoT Board, it is beneficial to verify that the cloud setup functions properly. Open the cloud function created in Section 1.2 Setting up a Cloud Function, and click on *View Logs*, as shown in Figure 1-10.

Figure 1-10. Showing Logs for a Cloud Function



In another tab, navigate to Dialogflow. In the upper right-hand corner, there is a text box labeled *Try it now*. Text entered here mimics what an actual voice command would be. For instance, type in *Could you print five?*. This sends a request to the Cloud Function. Figure 1-11 shows the result if everything went well. The first screenshot is from Dialogflow, and the second is from the Cloud Functions log view.

Figure 1-11. Testing the Cloud Setup With Dialogflow and Cloud Functions



1.3.3 Using a Google Assistant Enabled Device

Tip: If a device responds to a “Hey Google” prompt, it is most likely Google Assistant-enabled enabled and can be used for this example. Most Android® phones and Google Home devices are, for example, compatible.

A Google Assistant-Enabled Device can be used to issue voice commands with a “Hey Google” prompt. To use it with the cloud example created in the previous section, it must be logged into the same Google account. The procedure to log into a Google account depends on the device in question.

To use Google assistant, the dialog must be imported to *Google Actions* from Dialogflow. The data is imported by clicking the *“See how it works in Google Assistant”* link just below the *“Try it now”* text box in Dialogflow. This also redirects the user to the Google Actions console.

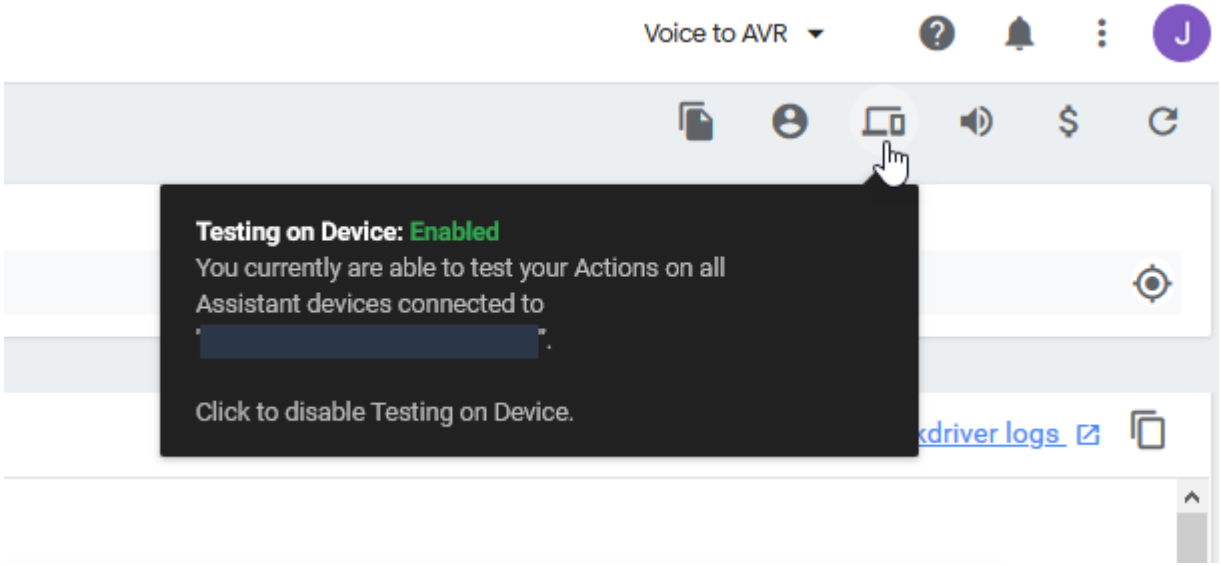
The action cannot be called before it has been assigned a name. A name is assigned by clicking on *"Develop"* in the bar on the top and entering a Display name. This example uses the name "Microchip Voice". Click on *Save* in the upper right-hand corner to activate the changes.

The setting *"Testing on Device"* must be enabled to use the device in question to invoke the command. Navigate to the *Test* tab on the top bar. In the upper right-hand corner, make sure *"Testing on Device"* is enabled, as shown in Figure 1-12.

The same voice command issued in 1.3.2 Testing and Verifying the Dialog (*"Could you print five?"*) can now be issued from the assistant device by the following dialog: ***Hey Google . . . Talk to Microchip Voice*** *- Here's the test version of Microchip Voice. Greetings! How can I assist?* - ***Could you print five?*** *- Printing 5 from the AVR-IoT Board*.

Tip: It is always possible to verify correct functionality by taking a look at the Cloud Function logs, as shown in Figure 1-10.

Figure 1-12. Enable Testing on Device



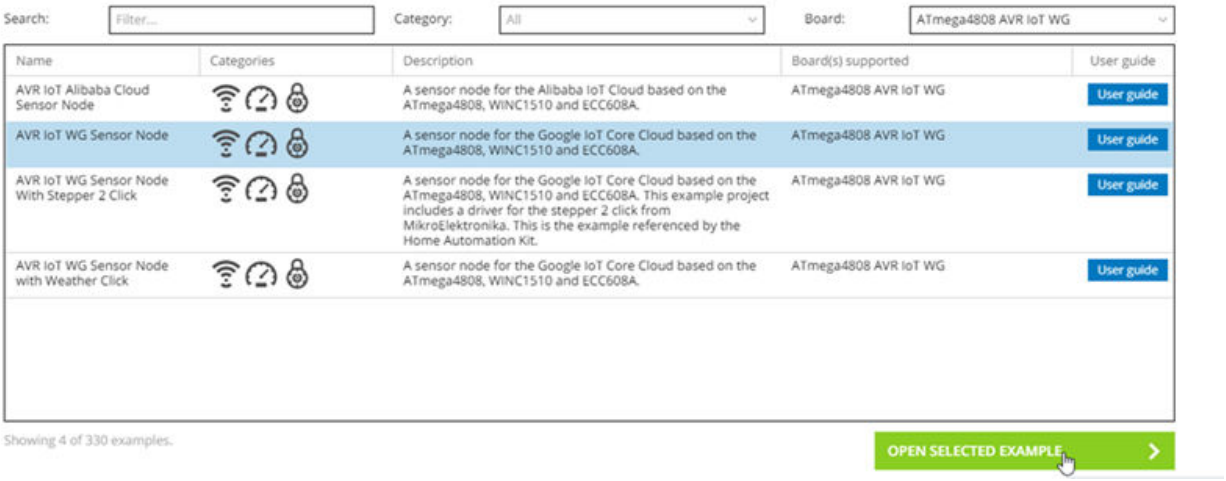
2. Adding the AVR-IoT WG Board

Tip: If the reader has problems following this section, it is recommended to read the AVR-IoT WG Board User Guide before proceeding. It can be downloaded here: http://www.microchip.com/DS50002809. A summary of key concepts is given when needed.

Tip: Atmel Studio can be obtained at https://www.microchip.com/mplab/avr-support/atmel-studio-7.

The final step for the print example is to configure the AVR-IoT Board to receive messages from the cloud and print the number to the computer. To import the AVR-IoT Boards firmware, open Atmel Studio and select File → New → Atmel Start Example Project. In the "Board" dropdown menu, find "*ATmega4808 AVR IoT WG*". In the filtered list, click on *"AVR IoT WG Sensor Node"*, followed by *"Open Selected Example"*, as shown in Figure 2-1.

Figure 2-1. Finding the IoT Board's Firmware in Atmel START



Under *"Cloud Configuration"*, enter the details as in Table 2-1. To import the source code to Atmel Studio, click on *"Generate Project"*.

Table 2-1. Cloud Configuration Entries for the IoT Board Firmware

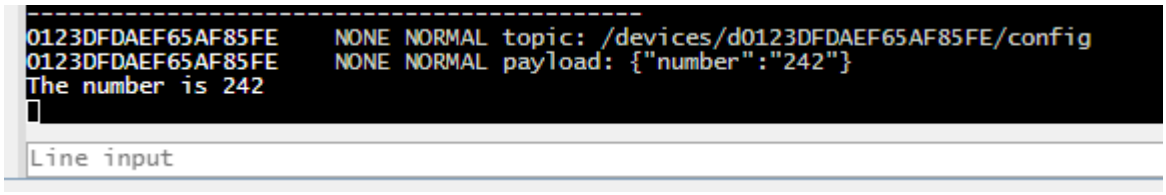
Name	Entry	Description
Project ID	voice-to-avr	The ID of the Cloud Project, defined in Section 1. Cloud Configuration. The ID can also be found on the front page of the Google Console, under *Project Info*.
Project Region	europe-west1	The region of the IoT Core, defined in Section 1.1 Configuring the IoT Core. It can also be found on the IoT Core Module front page.
Registry ID	voice-devices	The registry ID of the IoT Core, defined in Section 1.1 Configuring the IoT Core. It can also be found on the IoT Core Module front page.

.....continued		
Name	**Entry**	**Description**
MQTT Host	mqtt.googleapis.com	Which server to connect to for MQTT messages. Leave unchanged.

2.1 Handling Messages

In *main.c*, the function *receivedFromCloud* is called whenever a new message from the cloud arrives. With the cloud configuration described in Section 1. Cloud Configuration, all voice command messages are sent as a JSON string: *{"number": x}*. This JSON string must be parsed and the number *x* extracted. The extracted number *x* is then promptly sent to the PC through a *printf* call. See the source code below. By using a tool such as the *MPLAB® Data Visualizer*, the printed message can be viewed. With the board configured, whenever the user issues a print command through voice, the number appears in the Data Visualizer Terminal, as shown in Figure 2-2.

Figure 2-2. Resulting Number Printed in MPLAB Data Visualizer




```
// This handles messages published from the MQTT server when subscribed
void receivedFromCloud(uint8_t *topic, uint8_t *payload)
{
    debug_printer(SEVERITY_NONE, LEVEL_NORMAL, "topic: %s", topic);
    debug_printer(SEVERITY_NONE, LEVEL_NORMAL, "payload: %s", payload);

    char *numberToken = "\"number\":";
    char *subString;

    if ((subString = strstr((char *)payload, numberToken))) {
        uint8_t numberStrLength = 6;
        char numberStr[numberStrLength];

        // Start location of the number.
        char *currentChar = &subString[strlen(numberToken)] + 1;

        // As long as we do not hit a ", there are more digits. Record these
        uint8_t i = 0;
        while (*currentChar != '"')
        {
            numberStr[i] = *currentChar;
            currentChar++;
            i++;
            if (i > numberStrLength)
            {
                debug_printer(SEVERITY_WARNING, LEVEL_ERROR, "Number message has illegal
parameter: %s\n", subString);
                return;
            }
        }
        // Add the null terminator to make the string valid
        numberStr[i] = '\0';

        // Print the number
        printf("The number is %s\n", numberStr);
    }
}
```

3. Revision History

Doc. Rev.	Date	Comments
A	03/2020	Initial document release

The Microchip Website

Microchip provides online support via our website at http://www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to http://www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with

your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 978-1-5224-5724-4

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: http://www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820