Honeywell

ISA100 Gen X Radio Module

User's Guide

Release Independent

April 2018

Honeywell

Release Information

Document Name	Version No.	Release Number	Publication Date
ISA100 Gen X Radio Module User's Guide	1.0	Release Independent	April 2018

Contact Information

Contact: Phone:	Honeywell Technology Solutions Lab ACS Wireless COE Survey No. 19/2 Adarsh Prime Project Pvt. Ltd Devarabisanahalli, Varthur Hobli, Bangalore East Taluk, 91-80-26588360

Acronyms and Definitions

Acronym	Definition
DSSS	Direct-sequence spread spectrum
SPI	Serial peripheral interface
GND	Ground
EIRP	Equivalent isotopically radiated power
WCI	Wireless Compliance Institute
TCXO	Temperature Control Crystal Oscillator
SDR	Sensor Data Ready
CS	Chip Select
DAP	Device Application Process
DD	Device Description – a formal description of device objects used by host system tools.
DMAP	Device Management Application Process
UAP	User Application Process
AI	Analog Input
BI	Binary Input
UAPMO	User Application Process Management Object
APDU	Application Process Data Unit
RSSI	Received Signal Strength Indicator
RSQI	Received Signal Quality Indicator
EEPROM	Electrically Erasable Programmable Read Only Memory
LCD	Liquid Crystal Display
GPIO	General Purpose Input Output

Acronym	Definition
ASL	Application Service Layer
UDO	Upload Download Object
LPM	Low Power Mode
CRC	Cyclic Redundancy Code
PHY	Physical
DLL	Data Link layer
SAP	Service Access Point
NL	Network Layer
TL	Transport Layer
DPO	Data Process Object
DMO	Data Management Object
ISA	International Society of Automation

CONTENTS

Tables

Figures

1 Introduction

1.1 Overview

ISA100 is a wireless protocol standard defined for industrial automation for Process Control and related applications. The architecture of ISA100 is designed to be in coexistence with other wireless systems conforming to this standard, in addition to other networks operating at 2.4 GHz such as ZigBee™, WirelessHART™, 6LoWPAN, IEEE 802.11, Bluetooth™, and RFID systems. The Gen X ISA100 Radio Hardware Module is 2.4GHz band, 802.15.4 Radio based hardware, and can host multiple software protocol stacks like ISA100, zigbee™, 802.15.4, and WirelessHART™.

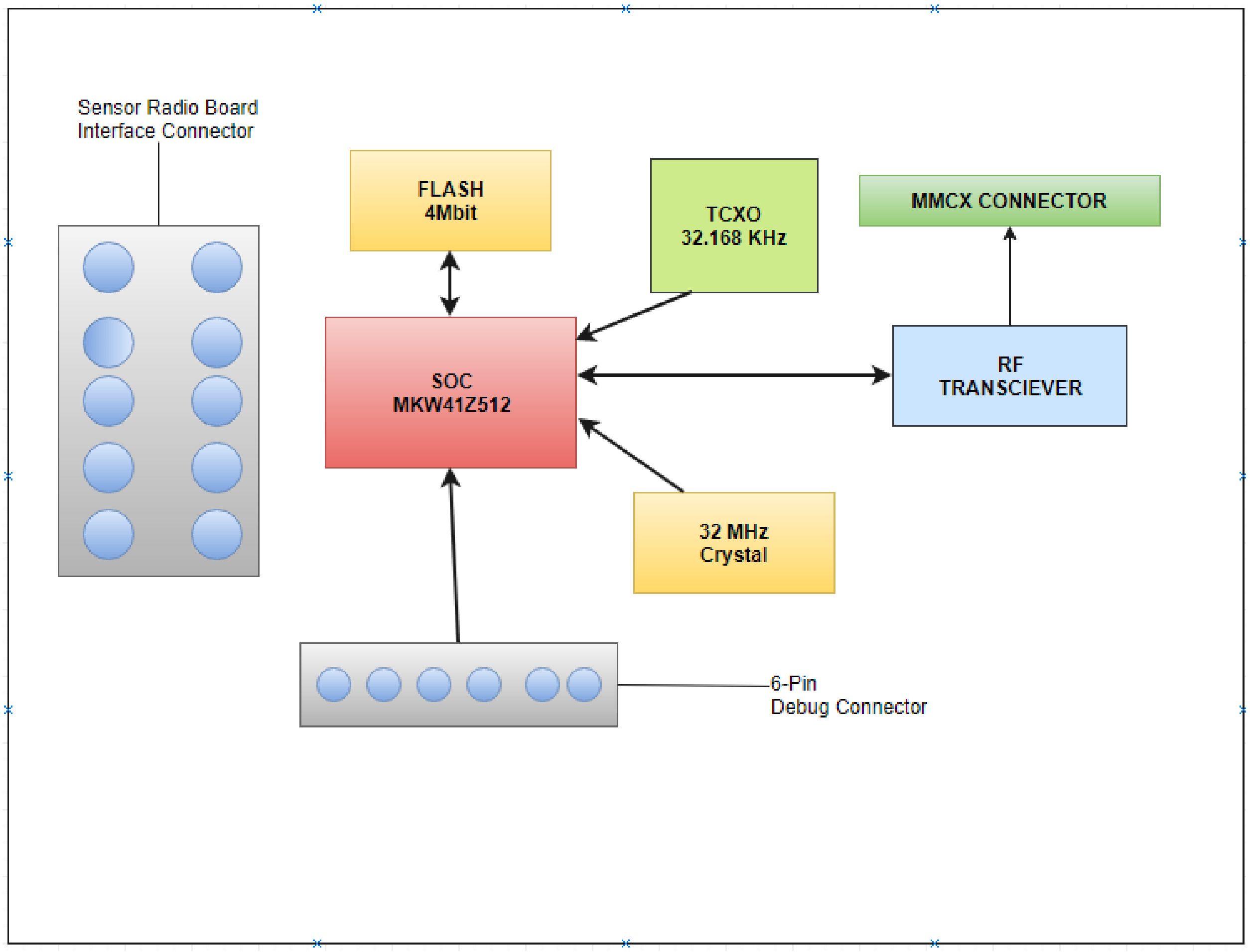


Figure 1: ISA100 Radio Board Block Diagram - Major Components

REFERENCE - INTERNAL:

For Compliance Statements, refer to Appendix A: Compliance Statements

For Agency Label Information, refer to Appendix B: Agency Label Information

2 Specifications

2.1 Features

1. The operational frequency ranges from 2.4 GHz to 2.483 GHz.
2. The maximum number of channels allowed is 16.
3. The channel spacing must be 5 MHz ((2405, 2410,…..2475).
4. The transmitted power varies from -5 dBm to +20 dBm (adjusted as per antenna).
5. The type of modulation technique used is direct-sequence spread spectrum (DSSS).
6. The data rate is 250 kbps.

2.2 Electrical requirements

1. The operating voltage ranges from 2.7 Volts to 3.6 Volts.
2. The operating temperature ranges from -40 º C to +85 º C.
3. The electrical consumptions are:
   - **Receive Mode (Rx):** 19mA
   - **Transmit Mode (Tx):** 80mA @16 dBm
   - **Average Sleep Mode:** 30uA (When Radio is not transmitting and receiving)

2. 3 Types of Antenna

The Antenna Connector used in the module is MMCX-J-P-H-ST-TH1, which is a Jack through-hole connector and mates with all the MMCX Plug (preferred from Samtec). It is mandatory to use a 50Ω connector towards the antenna for the described RF Power level performance.

The modular certification is performed for the antenna types (Refer to Table 1 Antenna Types). The certification is void if you use any other antennas than the ones mentioned in the table.

Table 1 Antenna Types

Sl.No	Antenna	Antenna gain	Power level settings
1	Centurion MAF94152 Omni Antenna	-2	16
2	L-COM HG2402RDR-RSP "Rubber-Duck"Omni Antenna	2	14
3	EM wave EM-B14503-MMP-Z6	4	14
4	L-COM HGV-2404U Omni Antenna	4	14
5	L-COM HGV-2409U Omni Antenna	8	10
6	L-COM HG-2414D remote Dish Antenna	14	6

2.4 Transmitter Power Configuration

Table 3 provides the transmit power output generated by the Radio at the input of the antenna connector present on the Radio board.

Table 2 Transmit Power Settings

S No.	Power Level
1	20dBm
2	19dBm
3	18dBm
4	17dBm
5	15dBm
6	14dBm
7	12dBm
8	11dBm
9	10dBm
10	6dBm
11	4dBm
12	3dBm
13	1dBm
14	0dBm
15	-1dBm
16	-5dBm

3 Connector Details

There are two connectors on the Radio board:

1. Debug Connector for programming or to debug the firmware on the board
2. Sensor-Radio interface connector

3.1 Debug Connector Details

The debug connector is used for programming and/or debugging the firmware on the board.

Debug Connector Pin Details

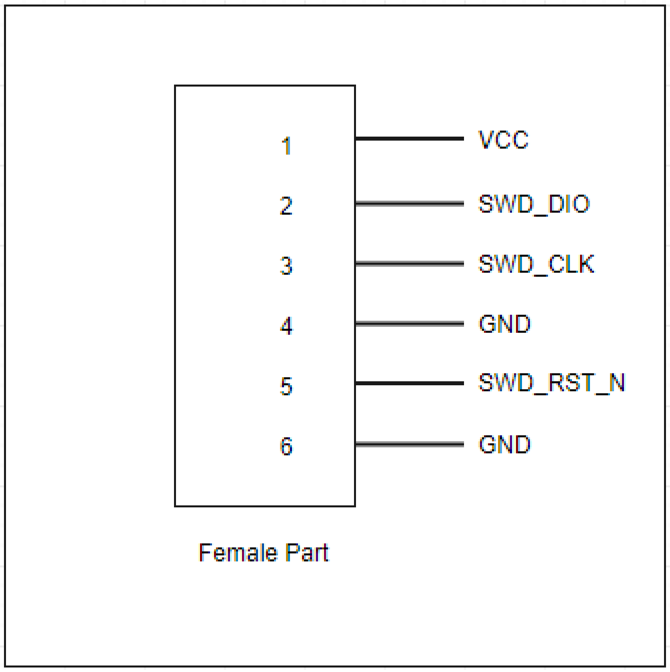


Figure 2: Debug connector pin details

SWS_DIO: data input/output
SWD_CLK: input clock
VCC: The operating voltage ranges from 2.7 Volts to 3.6 Volts.

Connecting the Debug Connector to J-Link JTAG Connector

The following diagram explains the connection details of the Debug Connector to J-Link JTAG Connector.

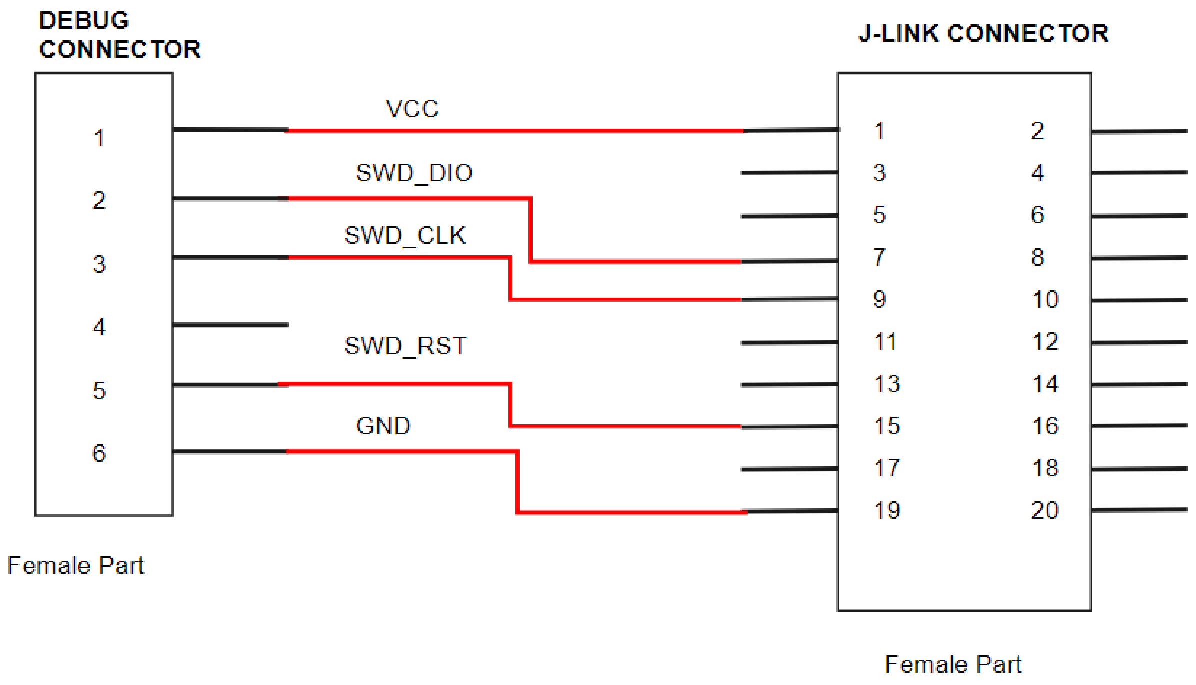


Figure 3: Connecting Debug Connector with J-LINK Connector

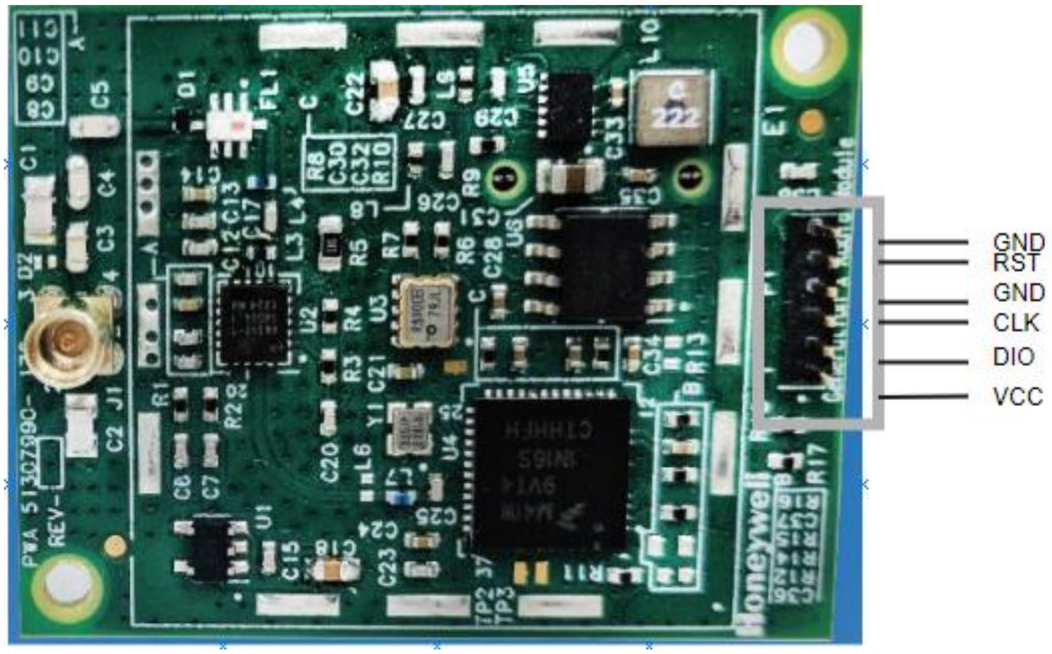


Figure 4: Debug Connector Pin Marking

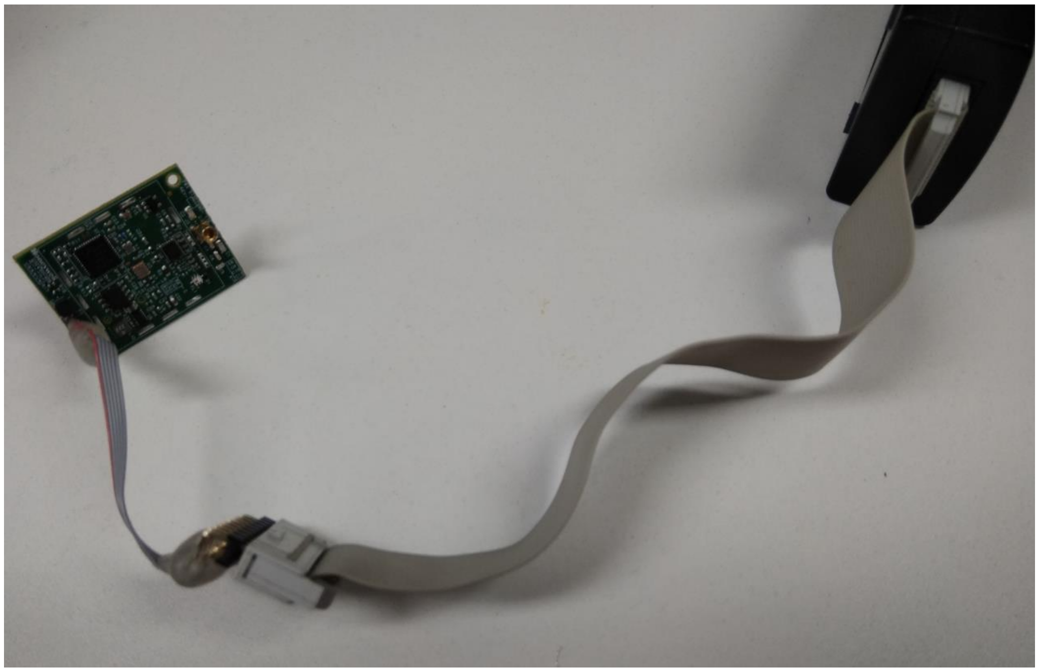


Figure 5: FET and Radio Board Connection

3.2 Sensor-Radio Interface Connector Details

The mating connector that connects the Radio interface is SFM-105-02-S-D-LC

Samtec.

MOSI, MISO, CS, and SCLK are used for SPI communication between the Radio and the Sensor, whereas GPIO is used as General Purpose Input/output as well as interruptible configurable pin.

Note: GPIO_02 pin in the Sensor-Radio interface connector is used as Sensor Data Ready interrupts (SDR).GPIO_01 is the inverse of Chip Select(SPI_CS).

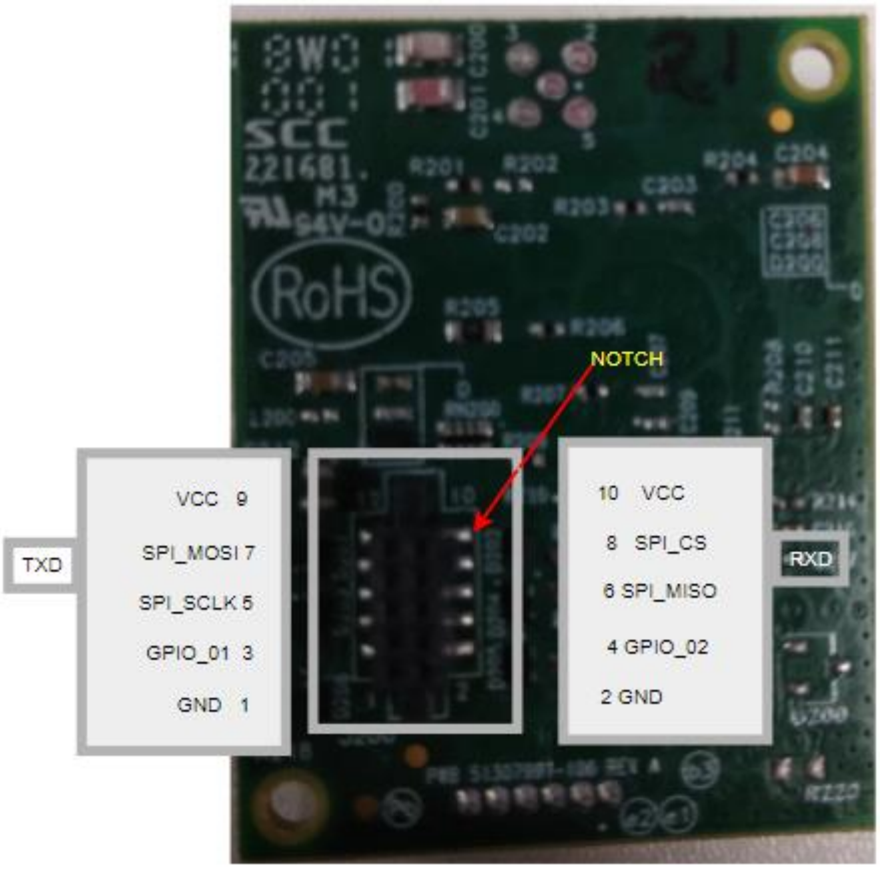


Figure 6: Sensor-Radio Interface Connector

The following table provides the Pin details of the Radio-Sensor interface connector.

Table 3 Pin Details of Radio Sensor Interface Connector

1 **GND**	**2** **GND**
3 **GPIO_01(**$\overline{\textbf{SPI\_CS}}$**)**	**4** **GPIO_02 (SDR)**
5 **SPI_CLK**	**6** **SPI_MISO**
7 **SPI_MOSI**	**8** **SPI_CS**
9 **VCC**	**10** **VCC**

4 Architectural Overview

The architecture of a device conforming to the ISA100 standards is described in terms of the OSI basic reference model. As shown in Figure 7: ISA100 Device Architecture, each layer provides a service access point (SAP). The services of a layer are defined as the functions and capabilities of that layer that are exposed through the SAP to the surrounding layers. The device manager is the entity within each device that performs the management function.

The ISA100 Gen X Radio Board implements the Device Manager (DMAP), ASLDE0 SAP, and TDSAP-0 and all the other subsequent layers. The Sensor processor board implements the User Application Process n (n = 2, 3, 4…15) and the equivalent ASLDE- n SAP and TDSAP-n.

The Sensor board usually contains one user application process. Within the user application process, you can find one or more transducer blocks, one concentrator block to publish data to the network, one upload download data object, and one user application process management object. If the Sensor board implements only one user application process, then the *n* is usually 2.

The sensor user application process communicates locally to the DMAP present on the Radio board using reserved local address methods. This includes a local contract id (0) for requests and reserved destination addresses defined by the Null Address Pointer that is used for the transfer of local data indications. Confirmation packets are referenced using a request handle that originates from the requesting data object.

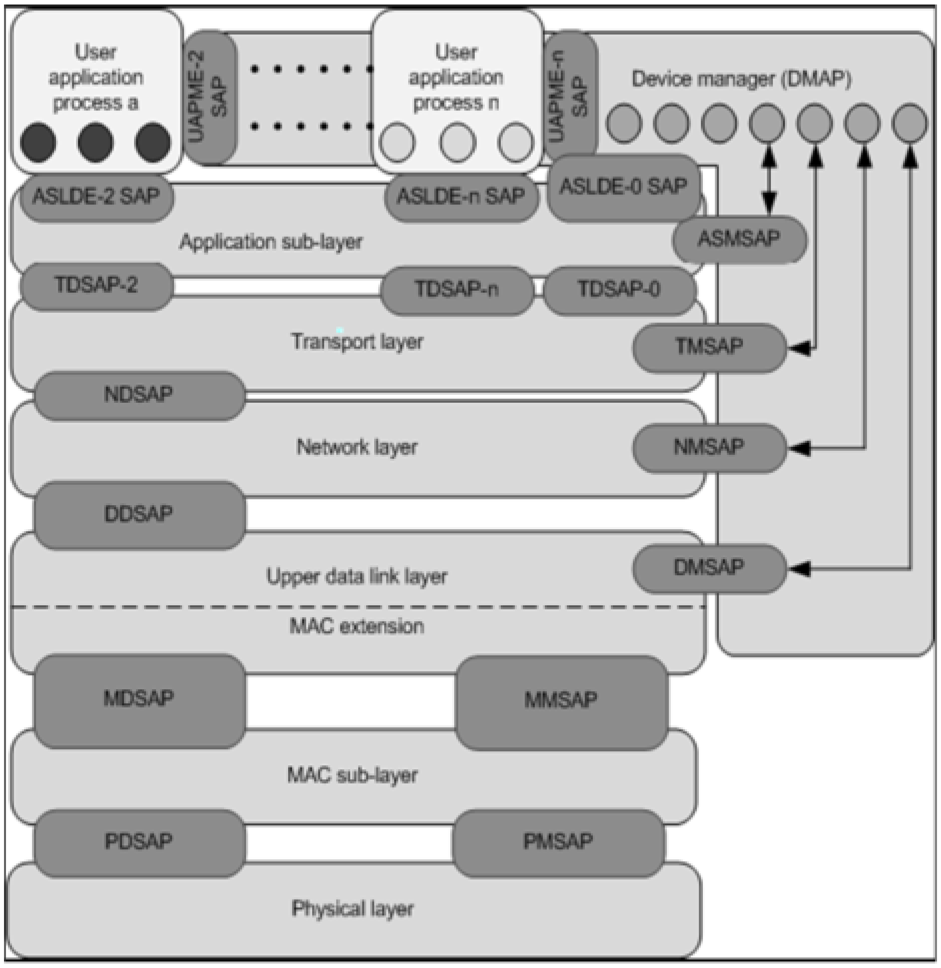


Figure 7: ISA100 Device Architecture

5. SPI Communication between the ISA100 Radio and the Sensor Board

5.1 SPI Configuration

SPI Inter-processor Communications (SPI_IPC)

The inter-processor interface between the Radio and the Sensor board is designed in such a way that the Radio board can treat the Sensor board as a peripheral device. The Radio is the SPI master and the Sensor board is the SPI slave. The SPI interface is not a query poll, but a bi-directional pipe that supports messaging between the application layers.

Data Rate

The baud rate of the SPI UART controls the data rate. The baud rate selected is arbitrary, as the SPI is a synchronous interface. For low power operation, the Radio SPI driver uses the 32768 Hz ACLK to generate the SPI clock; a divide by 2 is the minimum (fastest) clock rate that can be selected based upon the description in the MSP430 manual. This gives a bit rate of a baud rate of **15.384k baud**. This provides a byte data rate of 2048 per second. This rate assures that there is adequate time for the Sensor to process the SPI interrupts and maintain a reasonable packet throughput. An ISA packet of 42 bytes takes about 20 msec transfer. MSP430 ACLK is used so that the Radio and the Sensor can remain in the Low Power mode during data transfer as much as possible.

SPI Mode

The sensor operates in the SPI Mode with – CKPL = 0, CKPH = 1 (normally low clock, data clocked out on falling edge and clocked in on rising edge) using a four wire interface.

Arbitration

The Radio board is the SPI master and controls the transfer of data. The SPI interface is designed to be in low power so that it does not require periodic polling, a physical layer data, or headers to control the data transfer.

When the Radio needs to send data to the Sensor board (as in a request to a sensor block received from the wireless network), it initiates the transfer spontaneously, provided the SPI link is not currently in use. Packets can pass on the SPI link in both directions at the same time. Each packet is treated as an individual, that is, as an independent message that is passed to the appropriate application layer. An ISA100 packet transferred over the SPI is directed to the ISA100 application layer to be parsed and transferred to a local object or routed to a network or IR port.

The Sensor Board being the slave does not have the ability to initiate a communication exchange. Therefore, when the Sensor board has data to send to the Radio, it asserts the Sensor Data Ready (SDR) line as an interrupt to the Radio. When time permits, the Radio initiates a data transfer, and clocks the data in both the directions. The Radio board checks the type or length byte of the sensor message and continues to transfer SPI bytes (NULL bytes) until all the sensor message bytes are transferred.

The Sensor and the Radio can pass the data at the same time. However, this functionality supported by the SPI interface is not used in the current design.

Packet checking

A 2-byte CRC-16 is used in the SPI physical layer packet along with a 0x5A sync byte for checking the packet integrity.

Inter-packet delay

After a packet is transferred to or from the Sensor board, there is a lag, during which the Sensor reprioritizes its outgoing packets and processes any data that is received. During this time, the SPI is in an idle state. Radio places a fixed 10 msec delay between the packets to guarantee that the Sensor is ready for the next packet.

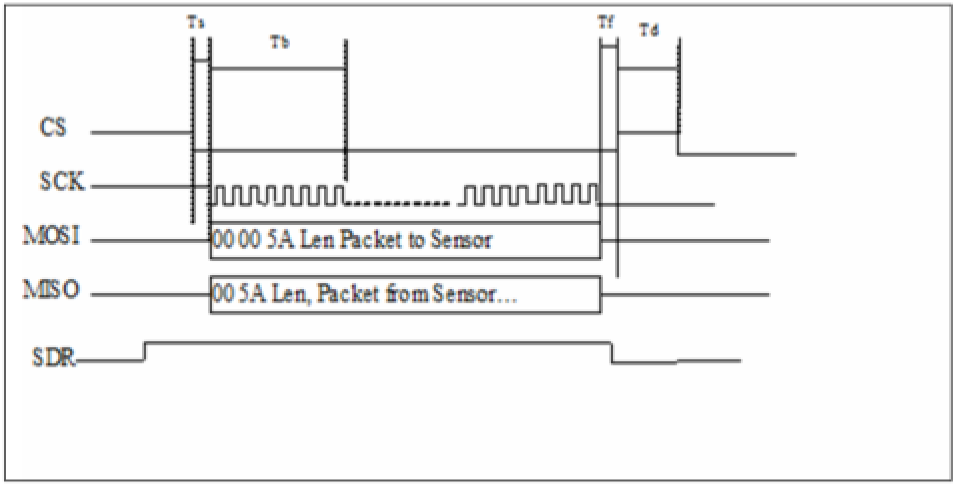


Figure 8: SPI Communication Signals

Parameter	Description	Minimum	Maximum
Ts	Time from CS active until start of first Master Byte transfer	0	-
Tb	Time for a Byte transfer	32768/(2 * 8) = 0.5 msec	-
Tf	Time from last bit sent until CS inactive.	0	-
Td	Time from CS inactive until next CS active.	10 msec	-

Packet Size and buffers

The SPI message definition allows for a 7-bit data length field. This limits the SPI message size to 127 bytes. However, the actual maximum message size is limited by the size of the SPI Rx buffer in the Radio board. In the ISA100 Radio board, the SPI Rx, and the Tx buffer size is only 80 bytes. Hence the Sensor board must not send packets exceeding this buffer size.

Interrupt processing

There are three interrupts on the Radio board that are used in the SPI interface.

Radio SPI interrupts

Sensor Data Ready (SDR)

The Sensor Data Ready interrupt is generated from the Sensor board to indicate that the data is ready to be sent to the Radio board. The Radio acknowledges this interrupt by reading the SPI packet from the Sensor.

SPI Rx Data

The Radio uses the SPI Rx data interrupt to receive the data. Each time the Master receives an Rx data interrupt, it reads the data from the SPI buffer and places a new byte to send in the SPI Tx buffer. The SPI Tx data interrupt is not required to allow a faster and lower power interface. As the SPI is buffered twice, this mechanism allows the SPI data to be transferred as a continuous bit stream.

This allows a minimal number of instructs cycles for processing these frequent interrupts and save significant power.

Chip Select Interrupt

Chip Select interrupt is generated by the Radio to indicate that the data is ready to be sent to the Sensor Board, or the Radio Board is ready to receive the data sent by the Sensor. The Sensor acknowledges this interrupt by turning on the SPI receive interrupt. When the data transfer is complete, the Chip Select line returns to its inactive state.

5.2 Inter-Processor Interface

The Sensor – Radio communication is achieved using a serial peripheral interface (SPI). Two

types of inter processor communication are supported:

1. **Network Packets** - Packets that have to be transmitted from the Sensor to the network.

2. **Local Packets** - Packets exchanged between the Radio and the Sensor.

The SPI IPC interface is designed efficiently to transport packets between the application sub layer for the Sensor application process and the application sub layer queue implemented on the Radio. Due to the ASL queue being located on the Radio board, the Sensor sends the request APDUs to the Radio. The Radio then sends the indication APDUs to the Sensor.

The Sensor board can send packets to the objects on the Radio as local requests packets. Local request packets are identified by the contract id 0. The Sensor board uses the general read of attributes from the Radio to retrieve information that is available as attributes in the DMO. The provisioning devices also use local data packets.

Contracts requests and other stack management functions are implemented as stack specific function calls that are implemented as a secondary protocol on the Sensor – Radio inter-processor communications. These stack management functions are provided for the use of a packet class header byte that directs the packet to stack management access interface.

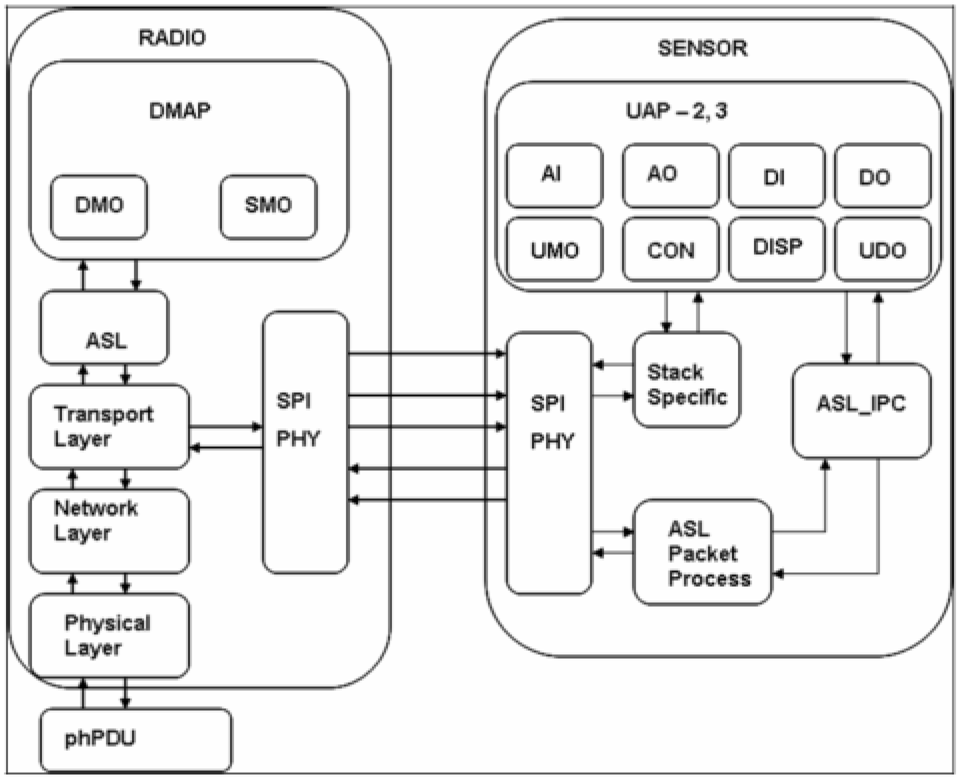


Figure 9: Sensor-Radio Inter-Processor Interface

5.3 Inter Processor Communication Packet description

SPI_IPC Physical Layer Packet Format

The SPI_IPC supports a simple physical layer packet that consists of a sync byte, a data length byte, and a CRC.

Sync Byte	IPC Data Length	IPC Data	CRC-16
5A	N	1-n Bytes	2 bytes

The SPI provides full duplex operation so that a packet can be sent from the Sensor at the same time a packet is being sent to the Radio. The SPI transfer starts by the Radio selecting the Sensor board using the Sensor select line. This select line indicates the start of a packet transfer to the Sensor. The Sensor then loads a 5A sync byte into the transmit register for the next data transfer. The Radio starts the data transfer when it receives a synchronization byte from the Sensor board.

When the Radio receives a 5A byte from the Sensor, it reads and sends a 5A back to the Sensor. When this second data transfer is complete, the next type received by the Radio is the number of bytes that the Sensor needs to send (in case sensor is the transmitter). The Radio then compares the number of bytes it has to send and proceed to transfer maximum number of bytes of the Sensor or Radio data.

When all the data bytes are transferred, the Radio de-asserts the Sensor select line to indicate the end of the packet. Then both the Sensor and the Radio inspects the received bytes for a correct CRC (discounting any received sync bytes.) If it is a valid message, the packet is passed to the inter-processor communication application sub layer interface.

SPI_IPC Data Format

The SPI_IPC application sub layer provides processing of valid messages that have been received over the SPI inter-processor interface. The inter-processor data processing utilizes a 4 byte header consisting of a packet class, packet type, packet ID, and packet length.

Packet Class	Packet Type	Packet ID	Packet Length	IPC Packet Data
1 byte	1 byte	1 byte	1 byte	1-n Data Bytes

Packet Class

Packet Class byte is an 8-bit value and defined as follows

4-bit	1-bit	3-bit
Packet Class	Request-Response	Reserved

The packet class is used for defining the basic construct of the packet. The two packet classes supported are: ASL packets and Stack specific.

The Request-Response bit is used only for Stack specific Class packets.

ASL queued packets class = 0x00

Stack specific class = 0x30 (Request) or 0x38 (Response)

Packet Type

The packet type is dependent on the packet class (refer to section . When the TLDE flushes the contract ID 0 from the ASL queue, it is converted to an ASL indication and it is sent to the Sensor board when it acquires its priority APDU.

ASL Queued Packet Types and Stack Specific Packet Types for details).

Packet ID

The packet ID identifies specific packets across the network. It checks for missed packets on the SPI interface.

Packet Data Length

The data length byte specifies the number of data bytes that are sent in this packet. You can check the packet length with the number of bytes transferred to verify whether the entire packet is transferred.

IPC Data Classes

The two packet classes supported are: ASL packets and Stack specific.
ASL queued packets class = 0x00

Stack specific class = 0x30 or 0x38

ASL Queued Packets

ASL queued packet class provides pass through access to the ASL queue located on the Radio board. This class of data packets allows prioritization and coordination of the Sensor packets with any Radio packets that are also queued. By using the ASL queue packets sent from the Sensor, the UAP can utilize concatenation and other functions that are not possible if simple Tx and Rx buffers are implemented and no ASL queue is implemented in the Sensor. In addition, ASL queue packets allow coordination of notification for contract errors as well as network retry.

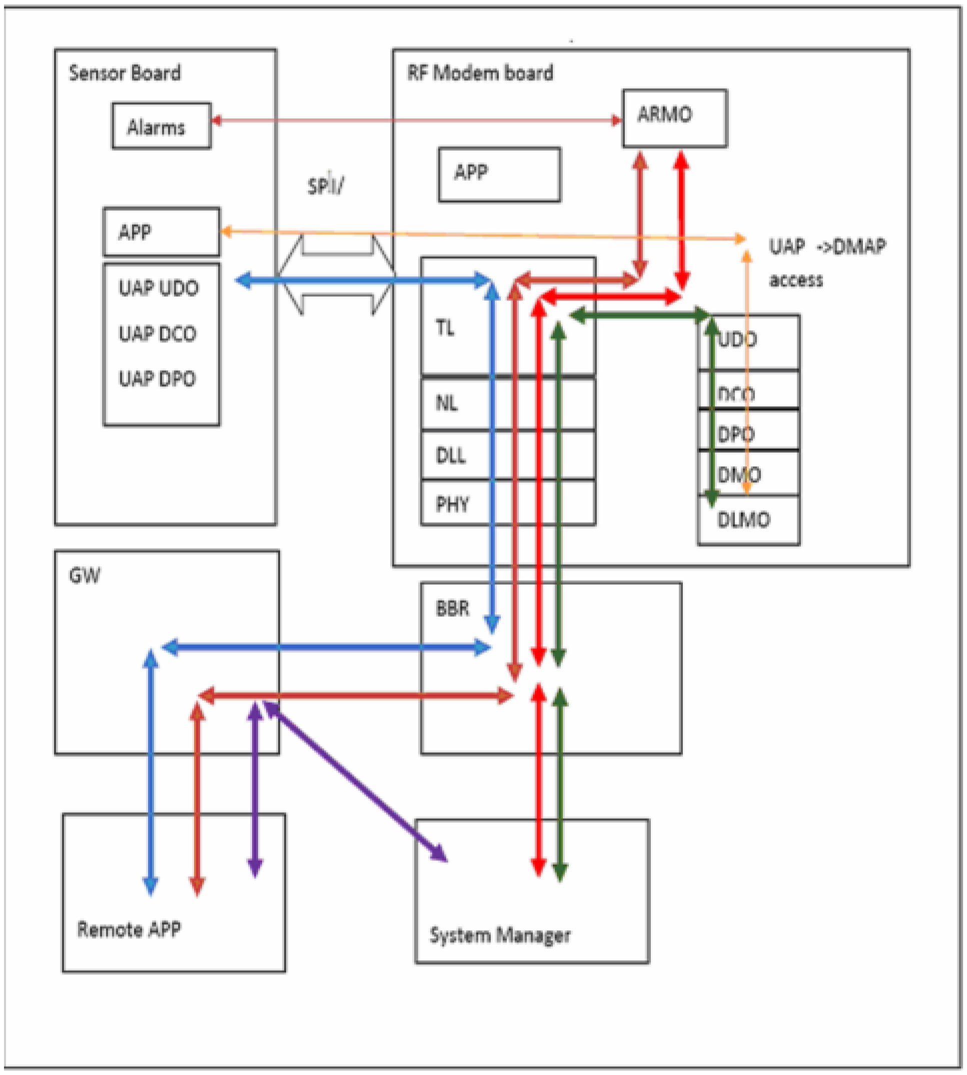


Figure 10: Sensor Board Integration with RF Modem using ISA100/Full API

ASL queued packets are used for the transport of local packets. When an ASL APDU request is received from the Sensor board, it is inserted into the ASL queue as a Tx packet. It is flushed from the ASL queue by the transport layer data entity (TLDE). The TLDE checks for a valid contract ID and, if valid, sends the packet down the stack to be transmitted on the wireless network. If the contract ID of an ASL APDU request is set to

0, the Radio TLDE understands that this is a local packet being sent to this device. The TLDE then changes the direction of the packet from Tx to Rx and sets the return network address pointer to NULL. Since the Rx and Tx packets are maintained in the same queue, the APDU need not be moved. You can modify only the ASL queue header of the Sensor packet.

The NULL return address for APDU requests is mapped to the contract ID 0 (local packets). This means that for the response to an APL, read request to the DMAP is placed into the ASL queue using contract ID 0. When the TLDE flushes the contract ID 0 from the ASL queue, it is converted to an ASL indication and it is sent to the Sensor board when it acquires its priority APDU.

ASL Queued Packet Types

The ASL queued class of packets supports three packet types: Request, Confirm, and Indication.

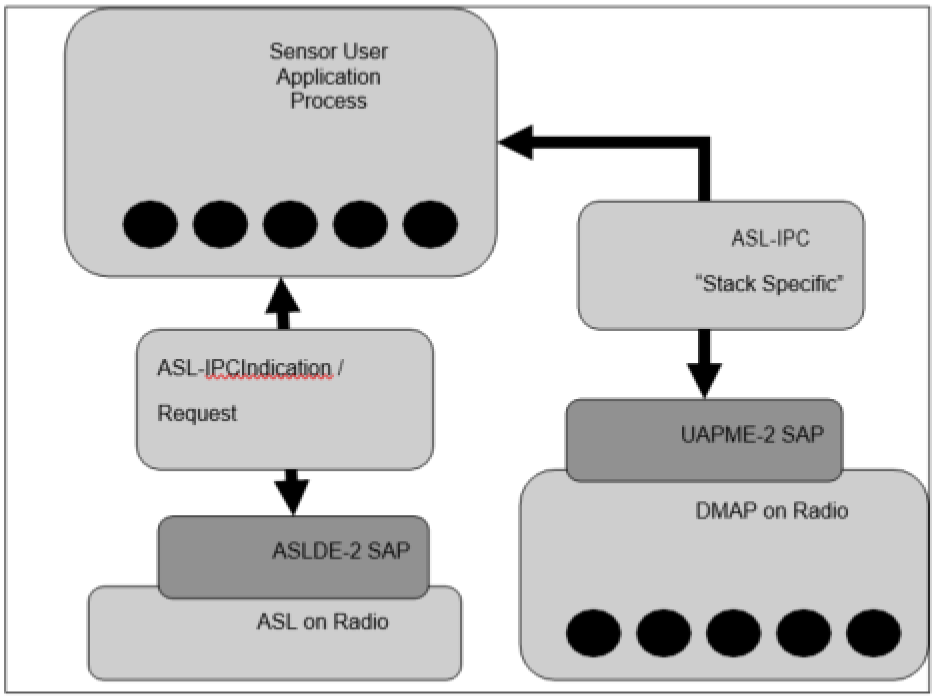


Figure 11: Sensor packets Logical Reference Model

Table 4 ASL Queued Packet Types

ASL Queue Packet Type	Type	Definition	Used
ASL Queued Indication	0	APDU indication pulled from ASL queue on the Radio board. APDU indications are marked as Rx packets in the Radio's ASL queue.	Yes
ASL Queued Request	1	APDU request to be inserted into the ASL queue on the Radio board. APDU request are marked as Tx packets in the Radio's ASL queue.	Yes

ASL Data Request

A data request packet format is sent from the Sensor to the Radio to request data to send to the ISA100 stack. The Sensor issues a data request packet to request read information from the Radio DMAP or to send a response to the Radio or to a network application that requested data from the Sensor. The Data Request format sends scheduled periodic publish data.

Priority	Dst TSAP ID	Src TSAP ID	Contract ID	App Data Length	App Data
UINT8	UINT8	UINT8	UINT16 Contract ID	UINT8	N Bytes

Contract ID 0 is reserved to allow the Sensor application objects to send local (non- network) request packets to the Radio. In addition, it allows the Radio application objects to send response packet back to the Sensor, using standard application interfaces without the need to establish formal contracts.

- The TSAP ID for the Sensor TSAP is 0x02 (0xF0B2).
- The TSAP ID for the IR provisioning TSAP is local 0x03 (0xF0B3).

ASL Data Indication

A Data Indication Packet indicates to the Sensor application process that an application data packet has been received. Local indications sent from the Radio board either is identified by all the 16 bytes of the address being 0. Local indications are typically marked by setting the address pointer to Null in the call to upper layer functions, or when inserted in the ASL queue.

Priority	**Src TSAP ID**	**Dst TSAP ID**	**Src Addr**	**App Len**	**App Data**
UINT8	UINT8	UINT8	16 BYTES	UINT8	N

Stack Specific Packet

The stack specific IPC data packets are used for non-ISA100 functions as mentioned in Stack Specific Packet Types. Addition of a stack specific function requires special handling functions, which in turn increases the flash and sometimes RAMS utilization. In addition, these functions are not prioritized in the ASL queue and can disrupt normal DMAP APDU message processing. For these reasons, this class of data has been only implemented and used if an operation cannot be achieved using the local support for ASL queued APDU services.

Stack Specific Packet Types

The Sensor uses ASL queued APDU class requests and APDU indications for both local and network APDU packets.

For Stack Specific Class, the following packet types are supported:

Table 5 Stack Specific Packet Types

Stack Specific Function	**Type**	**Definition**
ISA100_JOIN_STATUS	13	Read Request/Response for Join Status
ISA100_GET_TAI_TIME	15	Read Request/Response for TAI Time
ISA100_CONTRACT_REQUEST	21	Method for requesting contract from DMO
ISA100_NOTIFY_ADD_CONTRACT	22	Notification from DMO of new contract
ISA100_NOTIFY_DELETE_CONTRACT	23	Notification from DMO of deleted contract
ISA100_CONTRACT_TERMINATE	24	Notification from DMO to terminate contract
ISA100_NOTIFY_JOIN	27	Notification of change in Join Status

ISA100_GET_INITIAL_INFO	28	Get the initial information serial number and device tag descriptor from the sensor.
ISA100_ADD_UAP_ALARM	30	Add alarm message received from UAP to ARMO queue
ISA100_NOTIFY_RESET_CMD	120	Radio issues the command to reset the UAPMO
ISA100_NOTIFY_TAI_TIME	121	Radio sends the TAI time to the sensor board every 20 seconds
ISA100_NOTIFY_UAP_STATUS	129	Read Request /Response for the UAP status
ISA100_NOTIFY_POWER_STATUS	130	Sensor sends power supply status to radio
ISA100_NOTIFY_DEVICE_INFO	131	Sensor sends the serial number and device tag description to the Radio
ISA100_NOTIFY_ALARM_REGEN_BEGIN	140	Radio issues Alarm regen to sensor when system issues the Alarm recovery
ISA100_NOTIFY_ALARM_REGEN_END	141	Sensor generates Alarm Regen end when all pending alarms are generated
ISA100_NOTIFY_RSSI_RSQI	151	Sensor gets the RSSI and RSQI value from the device and updates in the LCS display

Initial Information Related Messages

ISA100_GET_INITIAL_INFO

On reset, the Radio requests the initial information from the Sensor as per the packet format in the following table.

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Checksum LSB	Checksum MSB
0x5A	0x04	0x30	0x1C	0x9C	0x00	XX	XX

ISA100_NOTIFY_DEVICE_INFO

The response from the Sensor to the Radio with initial information:

Synch Byte	Pkt Length	Pkt Class	Pkt Type	Pkt ID	Pkt size	Data Bytes	Checksum LSB	Checksum MSB
0x5A	0x2A	0x30	0x83	0x01	0x26	Description below	XX	XX

Data Byte 0	Data Byte 1	Data Byte2 -5	DataByte6-38
00	00	Device Serial No	Device Tag

Join Status Related Messages

ISA100_JOIN_STATUS:

The stack specific join status request is sent by the Sensor board to the Radio to know the network join status from the Radio. The Radio sends a response to the Sensor with the current join status. The request message has a data length of 0, whereas the response message has a 1-byte data value of join status.

Request for Join Status from Sensor

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Checksum	Checksum
0x5A	0x04	0x30	0x0d	0x01	0x00	XX	XX

Response from Radio

Sync Byte	Pkt Length	Pkt Class	Pkt Type	Pkt ID	Pkt size	Data	Check sum	Check sum
0x5A	0x05	0x38	0x0d	0x01	0x01	Join Status Value	XX	XX

The description of the Join Status is as follows.

Status	Value	Description
DEVICE_DISCOVERY	0	Device in discovery state: wait for an advertisement
DEVICE_RECEIVED_ADVERTISE	1	Received advertisement
DEVICE_FIND_ROUTER_EUI64	2	Request EUI64 from router
DEVICE_WAIT_ROUTER_EUI64	3	Waiting for EUI64 from router
DEVICE_SECURITY_JOIN_REQ_SENT	4	Security join request sent to security manager
DEVICE_SEND_SM_JOIN_REQ	5	Transient state
DEVICE_SM_JOIN_REQ_SENT	6	System manager join request sent
DEVICE_SEND_SM_CONTR_REQ	7	Transient state
DEVICE_SM_CONTR_REQ_SENT	8	System manager contract request sent
DEVICE_SEND_SEC_CONFIRM_REQ	9	Transient state
DEVICE_SEC_CONFIRM_REQ_SENT	10	Security confirmation message sent to security manager
DEVICE_JOINED	11	Device joined
DEVICE_NOT_JOINED	12	Device not joined (in back off)
DEVICE_NO_KEY	13	Device No Key

ISA100_NOTIFY_JOIN:

The Radio sends the join notification to the Sensor board on board/stack reset situations for the Sensor to know the status of the Radio immediately. This message is a notify message with the 1-byte join status in the data portion. There is no response message for this.

Sync Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Data	Check sum	Check sum

0x5A	0x05	0x30	0x1b	0x82	0x01	Joined - 01 Not Joined - 00	XX	XX

UAP status Related Messages

ISA100_NOTIFY_UAP_STATUS:

Radio can request for UAP status information from the Sensor .

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Checksum LSB	Checksum MSB
0x5A	0x04	0x30	0x81	0x01	0x00	XX	XX

Sensor sends UAP Status:

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Data Bytes	Checksum LSB	Checksum MSB
0x5A	0x08	0x30	0x81	0x01	0x04	Descriptio n below	XX	XX

Data Byte0	Data Byte1	Data Byte2	Data Byte3
00	00	No of User Applications	Status of Application

ISA100_NOTIFY_RESET_CMD

This command is sent from the Radio to the Sensor, and the Sensor resets the UAPMO based on the Type of reset (WARM, HARD, or FACTORY DEFAULT).

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Data Byte0	Checksum LSB	Checksum MSB
0x5A	0x05	0x38	0x79	0x01	0x01	Reset Type	XX	XX

Periodic Messages:

ISA100_NOTIFY_TAI_TIME

The Radio sends the TAI time update every 20 seconds to the Sensor board. This message contains 10-bytes data, 6-byte TAI time, and 4-byte reserved. The TAI time is a 6-byte value, 4-byte seconds, and 2-byte fraction of a second (2^{-15}). All the data is in big endian format.

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Data1-Data6	Data7-10	Check sum	Check sum
0x5A	0x0E	0x38	0x0f	0x01	0x0A	6 bytes of TAI time	Rsrvd	XX	XX

ISA100_GET_TAI_TIME

Sensor can request for TAI time if required. Usually the radio sends the TAI time every 20 seconds.

The stack specific Get TAI Time request is sent by the Sensor board to the Radio to know the current TAI time. The Radio sends a response to the Sensor with the current TAI time in seconds. The request message has a data length of 0, whereas the response message has 4-byte data value of TAI time.

Sync Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Check sum	Checksum
0x5A	0x04	0x30	0x0f	0x01	0x00	XX	XX

ISA100_NOTIFY_POWER_STATUS

This is a notification message sent by the Sensor periodically to the Radio board to update the power supply status. The message contains 1-byte power supply status and the values are as follows:

Sync Byte	Pkt Len	Pkt Class	Pkt Type	Pkt ID	Pkt size	Data	Data	Data	Check sum	Check sum
0x5A	0x07	0x30	0x82	0x01	0x03	0x00	0x00	Pwr State Valu e	XX	XX

States	Value	Description
PWR_STATUS_LINE	0	line powered
PWR_STATUS_BATTERY_1	1	battery powered, > 75%
PWR_STATUS_BATTERY_2	2	battery powered, between 25% and 75% remaining capacity
PWR_STATUS_BATTERY_3	3	battery powered, < 25%
PWR_STATUS_UNKNOWN	255	status not known yet

ISA100_NOTIFY_RSSI_RSQI

This command is sent from the Radio to the Sensor with information on RSSI and RSQI values to be displayed on the LCD.

Sync Byte	Packe t Lengt h	Packe t Class	Packe t Type	Packe t ID	Packe t size	Dat a Byt e 0	Dat a Byt e 1	Checksu m LSB	Checksu m MSB
0x5A	0x06	0x38	0x97	0x01	0x02	RSSI	RSQI	XX	XX

Contract Related Messages:

ISA100_CONTRACT_REQUEST:

This message is a request message sent by the Sensor board to the Radio board to request for a new contract. The data contains the contract message as defined in the ISA100 standard.

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Data0-Data36	Check sum	Check sum
0x5A	0x29	0x30	0x15	0x01	0x25	Description Below	XX	XX

Data0	Req Id
Data1	Contract Req Type Note: Contract Req Type van is New/ Renew/ Modify.
Data2	Service Type: Periodic/ Non Periodic
Data3, Data4	Source TL Port: 0xF0B2
Data5- Data20	Remote Address(16 Bytes)
Data21, Data22	Destination TL port
Data23	NA
Data24-Data27	Contract Life
Data28	Contract Priority
Data29, Data30	Max packet Size
Data31	Contract Reliability

Periodic Contract Request

Data32,Data33	Data34	Data35,Data36
Contract BW Period	Contra ct BW Deadline	Contract BW Phase

Aperiodic Contract Request

Data32,Data 31	Data34,Data35	Data36
Contract BW commit Burst	Contract BW Excess Burst	Contract BW Window

ISA100_NOTIFY_ADD_CONTRACT:

This message is a request message from the Radio to the Sensor board to notify the contract response to the Sensor when the contract response is received from the system manager. The data contains the contract response as defined in the ISA100 standard.

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Data0-Data17	Check sum	Check sum
0x5A	0x16	0x38	0x16	0x01	0x12	Description Below	XX	XX

Data0	Data1	Data2, Data3	Data4	Data5-Data8	Data9	Data10	Data 11	Data12
Contract Req ID	Contract Status	Contract ID	Service Type Periodic /Aperiodic	Expiry Time	Priority	Contract Max APDU size	NA	Contract Reliability

Periodic Contract

Data13,Data14	Data15	Data16,17
Contract BW Period	Contract BW Phase	Contract BW Deadline

Aperiodic Contract

Data13,Data14	Data15,Data16	Data17
Contract BW Commit Burst	Contract BW Excess Burst	Contract BW Window

ISA100_CONTRACT_TERMINATE

This message is sent by the Sensor board to the Radio board to delete an existing contract. The data contains the contract termination message, which is 2-byte Contract Id.

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Data0, Data1	Data2	Check sum	Chec k sum
0x5A	0x07	0x30	0x18	0x01	0x03	Contract ID	Operation- Terminate 0/ Deactivate1/ Reactivate2	XX	XX

ISA100_NOTIFY_DELETE_CONTRACT

This message is sent by the Radio board to the Sensor board to delete the contract once the terminate contract confirmation is received from the system manager. The data contains the contract delete message, which is 2-byte Contract Id.

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Data0, Data1	Check sum	Check sum
0x5A	0x06	0x38	0x17	0x01	0x02	Contract ID	XX	XX

Alarm Related Messages

ISA100_NOTIFY_ALARM_REGEN_BEGIN

The Radio issues Alarm regen begin message to the Sensor board when the system issues the Alarm recovery to the Radio board. The Sensor board generates all the pending alarms when this message is received. There are 4 such alarms generated each for 4 message categories (communication, process, device diagnostics, and security). The 1- byte data value carries the alarm category.

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Data0	Check sum	Check sum
0x5A	0x05	0x38	0x8C	0x01	0x01	Alert Category	XX	XX

ISA100_NOTIFY_ALARM_REGEN_END

The Sensor generates alarm regen end message to the Radio board, one per category of alarms basis, when all the pending alarms are issued. The 1-byte data value carries the 1- byte category value.

Note: The Radio board also sends an acknowledgement to the sensor when it receives the alarm regen end message.

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Check sum	Check sum
0x5A	0x04	0x30	0x8D	0x01	0x00	XX	XX

ISA100_ADD_UAP_ALARM

This message is generated by the Sensor board to send an alarm to the system. This message is specially treated from the other ASL message because this message needs to be added to ARMO queue present in the Radio board. The data contains the standard alarm message as described in the ISA100 standard.

Synch Byte	Packet Length	Packet Class	Packet Type	Packet ID	Packet size	Data Bytes 0-N	Check sum	Check sum
0x5A	0x0F	0x30	0x1E	0x01	0x0B	Description Below	XX	XX

Byte 0	Byte 1, Byte 2	Byte3	Byte4	Byte4	Byte5	Byte6	Byte7	Byte8-ByteN
UAP ID	SOID	Alert Class	Alert Direction	Alert Type	Alert Priority	Alert Category	Alert value length	Alert Value

6. Message Flow between the Radio and Sensor

6.1 Start-up Mode

Radio **On Power ON** (or) Restart, requests for some initial information from the Sensor board, using stack specific message ISA100_GET_INITIAL_INFO (0x1C). The Sensor responds with ISA100_NOTIFY_DEVICE_INFO (0x83 command) section.

The Radio sends the ISA100_GET_INITIAL_INFO message continuously once every 1.6 seconds up to 50 seconds. During this time, the Radio expects the Sensor to send at least battery power status using ISA100_NOTIFY_POWER_STATUS (0x82) command. If the Sensor does not send ISA100_NOTIFY_DEVICE_INFO and ISA100_NOTIFY_POWER_STATUS packets to the Radio in 50 seconds, the Radio assumes low battery power status and proceeds further. After 50 seconds, the Radio goes to a normal mode condition, and then the Sensor can send ISA100_JOIN_STATUS (0x0D) command Radio to request for join status.

6.2 Normal Mode

In addition to initial notification of Power Status using ISA100_NOTIFY_POWER_STATUS (0x82), the power status information have to be communicated to the Radio periodically every minute or on a change of power supply status or both.

The Sensor has to keep querying for the join status every 30 seconds (ISA100_JOIN_STATUS: (0x0D)) until it receives the ISA100_NOTIFY_JOIN (Join Status Data is 1) from the Radio, informing the Sensor that the Radio has joined the network. When the Radio drops off from the network, the Sensor is notified by the Radio by the same message ISA100_NOTIFY_JOIN (Join Status Data is 0) and sensor resumes its request for join status.

For the Sensor to publish data over the network, it requires a contract. The Sensor has to request for a contract using ISA100_CONTRACT_REQUEST. It takes a while for the Radio to receive the contract details from the system manager. The Sensor keeps requesting for a contract ISA100_CONTRACT_REQUEST every 5 seconds until the ISA100_NOTIFY_ADD_CONTRACT message containing the contract details is received from the Radio. Once the contract is details are received the AI and BI data will be published over the network as per the publish period obtained from the contract details message. The Radio also keeps the Sensor notifying the TAI time every 20 seconds and the RSSI and RSQI values every 1 minute.

The Sensor is also expected to transmit the alarms if any, and also notify the change in state of alarm using ISA100_ADD_UAP_ALARM

The system manager can request for an alarm re-generation using ISA100_NOTIFY_ALARM_REGEN_BEGIN if necessary, and the Sensor is expected to send all the active alarms. When all the alarm information is sent, it has to send an alarm regeneration end message, ISA100_NOTIFY_ALARM_REGEN_END.

7. Sample Code Description

7.1 Scheduler

Sl No	File	Description
1	Main	The Sensor firmware has a round robin scheduler for • Processing IR events • Processing SPI received messages , timed functions- Sensor • UAPMO and Display tasks every 1 second • Alarms every 2 minutes, • Power status every 1 minute • Query power status every 5 seconds until the device joins and then puts the device to sleep.

7.2 Interrupts

SL No	Interrupts	Interrupt Description
1	Timer Interrupt	Occurs every 250 msec
2	SPI chip select interrupt	Selects the Sensor for SPI transmission
3	SPI Receive Interrupt	On reception of a Data Byte
4	IR Interrupt	IR reception or time outs

7.3 ISA100 Objects

The sensor firmware implements 5 objects:

Table 6 Five Objects of the Sensor Firmware

Sl No	Files	Description
1	UAPMO Object	Functions for: • UAPMO initializations • Read UAPMO attributes • Write to UAPMO attributes, request and manage device diagnostic alerts • Request, activate, delete, and terminate contracts
2	Concentrator Object	Functions for • Concentrator initializations • Read concentrator attributes • Write to concentrator attributes • Execute requests • Publish data and and requests • Activate, delete, and terminate contracts
3	AI Object	Functions for • AI initializations • read AI attributes • Write to AI attributes • Execute requests and assemble data to be published

4	BI Object	Functions for • BI initializations • Read BI attributes • Write to BI attributes • Execute requests and assemble data to be published
5	UDO Object	Has functions for • UDO initializations • Read UDO attributes • Write to UDO attributes • Execute requests and functions supporting Data Download

7.4 Device Drivers

SL No	Files	Function Description
1	SPI_IPC	Handles SPI0 initializations, SPI receive Interrupt, SPI receive, and transmit processing.
2	SPI_Sysio	Handles SPI1 Initializations, SPI read and write functions, SPI enable and disable Chip select.
3	IR_driver	Handles IR initializations, IR interrupts, receive and transmit processing
4	EEPROM driver	Functions for EEPROM read, write, compare, unprotect, and read status
5	LCD Driver	Functions for LCD init string display, Display for RSSI and RSQI

7.5 Alert Configuration and Reporting

All the device diagnostic errors are classified into four categories:

1. Maintenance Alert
2. Out Of Spec Alert
3. Function Check Alert
4. Failure Status Alert.

Table 7 describes how the detailed device status gets mapped into four Broad Alert types.

Table 7 Detailed Device Status

Alert Descriptors for Configuring Alerts via UAPMO Attributes (Attribute No)	**General Diagnostic Categories (UAPMO.DEVICE_STATUS attribute 67)**	**Device Detailed Status (UAMPO attribute 102)**
failure_status_alert_desc (108)	FAULT_IN_ELECTRONICS	DEV_ST_RADIO_ERR DEV_ST_ELEC_FAIL DEV_ST_ROM_FAULT DEV_ST_RAM_FAULT DEV_ST_NVM_FAULT DEV_ST_AD_FAULT
	FAULT_IN_SENSOR_ACTUATOR	DEV_ST_INPUT_FAIL
	SOFTWARE_UPDATE_INCOMPLETE	None
function_check_alert_desc (107)	OUT_OF_SERVICE SIMULATION_ACTIVE SENSOR_DATA_UNCERTAIN	None
out_of_spec_alert_desc (106)	INSTALLATION_CALIBRATION_PROBLEM	DEV_ST_CAL_ERR DEV_ST_CHAR_FAULT DEV_ST_EXCESS_ZERO DEV_ST_EXCESS_SPAN DEV_ST_EXCESS_CAL
	OUTSIDE_SENSOR_LIMITS	DEV_ST_MB_OVT DEV_ST_MB_OVL
	ENVIRON_CONDITIONS_OUT_OF_SPEC	None
maintenance_alert_desc (105)	POWER_CRITICALLY_LOW	DEV_ST_LOW_BAT DEV_ST_LOW_EXT_PWR
	FAULT_PREDICTED	None
	POWER_LOW	None
	SENSOR_MAINTAINACE_REQUIRED	None

UAMPO Attribute 67 (DEVICE_STATUS) broadly gives the details about the failure in the device, and the details about that particular failure is provided in the UAMPO attribute 102 (DEVICE_STATUS_DETAIL).

For Example, when FAULT_IN_ELECTRONICS bit is set in the UAPMO.DEVICE_STATUS attribute, the particulars about the Electronics fault (ROM, RAM, or NVM failure and so on) are given in the APMO.DEVICE_STATUS_DETAIL attribute.

There are four alarms generated by the device.

1. failure_status_alert
2. function_check_alert
3. out_of_spec_alert
4. maintenance_alert

Each of these four alerts is enabled/disabled using four independent alert report descriptors specially defined in the UAPMO (attribute 105, 106, 107, and 108).

Each of the Alert report descriptor has two elements, each of size 1 octet.

- Alert report disabled
- Alert report priority

Each of the Alert is of size 1 octet and is user configurable. The Alerts can be enabled or disabled by clearing or setting the **Alert report disabled** element of the **Alert Descriptor** attribute.

A single byte is used for controlling the alert descriptor in the code. As the **Alert report disabled** is a Boolean internal to the code, a single bit is allocated for this purpose. The Alert report priority varies from 0 to 15. So a nibble is allocated for this. The rest of the 3 bits are used for internal purpose to maintain the state of the alarm (alarm sent or not sent, alarm is in-alarm, or returned to normal condition, and so on).

An example alarm generation code is placed in the sample code to show the generation of the alarm and to reset the alarm back to normal.

The current code sets and resets Input Fail Alarm (Failure Status Alert) every 2 minutes implemented through the function **send_alarm()**. The Function **uapmo_set_status_detail_1()** called in the main.c file sets and resets the particular bit of the UAPMO.DEVICE_STATUS_DETAIL to reflect the status of the alarm. This variable gives the details about the device diagnostics.

The function **uapmo_cycle(),** which is called in main.c file, sends the alarm to the Radio and is called every 1 second.

The function **uapmo_update_diagnostic_status ()** maps the detailed device diagnostics (column 3 of table) to the broader 4 categories (column1 of table) and updates the details into the uapmo device_status variable.

When the device joins the network, the function **uapmo_alert()** function checks if the corresponding alert is enabled or disabled by checking the uapmo_param_desc descriptor table by passing the UAPMO parameter numbers refer column1 of Table 7 Detailed Device Status. If the Alert is enabled , the device status and the alert diagnostic bit are set, and the device is not in alarm, the parameter value of the UAPMO alert descriptor variable gets changed to indicate the device is in alarm and that there is a change in the alert status. The stack message is sent from the Sensor to the Radio to indicate the status of the alarm.

The function **uapmo_send_diag_alert()** is called for sending the alarm .This function internally calls **wapp_alert_req()** for transmitting the alarm.

Every Alarm/Event consists of the following fields:

- UAPMO object ID (or if an AI object generates the alarm, that object ID will come).
- Alert Class – Event/ Alarm;
- UAPMO Diagnostic Alert types – Failure Status/Out of Spec/ Maintenance/function check
- Alarm Direction: In Alarm/Return to Normal(No Alarm)
- Alert Category: Device Diagnostic/ Process/ Security/Communication Diagnostic
- Alert Priority
- Alert Value

When the Input Fail Alarm goes off, the Sensor again sends the message to indicate the change in Status.

An Alert is of two types, an **Alarm** or **Event**. This is indicated by the Alert class. An Event can be a device restart or Firmware upgrade failed, and so on. An example Event generation when device is restarted is given in the function **uapmo_alert().**

Alert Type notifies the type of alarm that you are generating. Each Alert is given a number starting from 0 to 255. The Alert types must be defined in the Attribute ID 101 of each object within the DD to convert these numbers to strings to show in the User Interface. Attribute 101 does not exist in the device. It exists in the DD file for alert type to string conversion for user visible strings.

Alert Categories are of four types. Generally, the Device Diagnostics and process alerts are generated or handled by the Sensor board and the security and communication alerts are handled or generated by the Radio.

Alert Direction is used for alarm to indicate alarm occurrence or return of the alarm to normal condition.

Alert priority helps to prioritize alerts when multiple alerts occur at the same time and prioritize the one to be sent first.

A value is associated with each alert to indicate the value related to that alert. For example, a battery low voltage alarm can carry the current battery voltage in the alert, and so on.

Alarm Regeneration

This is a separate functionality to recover the active alarms in the Sensor. This message is sent when the system manager requests an alarm regeneration if it loses all the alarms by database corruption. The alarm regeneration request comes as a stack packet and sets the UAPMO variable uapmovar.alert_regen. The function uapmo_alert, which gets called every 1 second, checks all the alarms in the alert descriptor table for active alarms and sends the corresponding alarm messages. When all the messages are transmitted, it sends the stack message ISA100_NOTIFY_ALARM_REGEN_END.

IR Communication Interface

REFERENCE -EXTERNAL

For details about IR physical layer and communication details, refer WCI ISA100.11a Out of Band Provisioning Specification document.

IR Communication interface works at 9600 baud rate. The IR communication is implemented using Timer A for bit detection and generation. The IR interface is designed to normally operate in a very low power listen mode where it turns off the IR transceiver and wakes every 1 second and enables the IR receiver for 3.5 character times to listen for a wakeup bit stream. Detection of 2 pulses in 3.5 character times is taken as a valid wakeup pulse and puts the micro in LPM1 mode so that the SM clock is running, and further data can be successfully received. If less than 2 pulses are detected within 3.5 characters, then it is treated as no communication is happening on the IR port and IR port is power down. If a wakeup bit stream (> 2 pulses) is detected, the IR interface goes to the receive mode. When in receive mode, the interface remains in the receive mode for a 10 second time-out or with no incoming data. If an incoming data is received within 10 seconds time-out, the time-out is reinitialized back to 10 seconds. Once the 10 second time-out has occurred without receiving any incoming data for 10-seconds, the IR driver returns to low power listen-mode where the IR transceiver is switched off. 0x C0 is the synchronization byte for reception and transmission. A train of 1000 FFs are used for waking up the device from sleep.

A total of 10 bits is transmitted for a byte. Start bit is always 0 and the Stop bit is 1. “0” is represented by a pulse of the duty cycle 3/16 and “1” is represented by the absence of a pulse.

The function ir_do_ events() is called in the round robin scheduler for processing the IR events based on the IR event flag. Besides the 1 second and 10 second time-outs, three interrupts are used for IR transmission and reception.

1. Capture event for edge detection – start of a byte reception
2. Bit time-out interrupt for bit detection or transmission.
3. Inter-byte time-out for packet detection

IR reception

The function **IR bit timer** is called on bit timer expiry and is used for assembling the bits to form a byte for reception or transmission. The Start bit is detected by a capture event (since it is zero always) and from then the bits are assembled till Stop bit. Timer expiry is used for identifying the end of bit period, and capture interrupt flag is used for identifying if the received bit is a “0” or “1”. Once the whole byte is assembled, the timer is set for 3.5 character time expiry, that is, the next byte is expected to arrive within 3.5 character times. Once the synchronization byte (0xC0) is received, the further bytes are considered as valid. The received data bytes are stored in a buffer. On expiry of 3.5 character timer, the flag IR event done is set, which indicates either a fully received packet or reception time-out. When the IR packet is fully received, it is sent for further processing based on the source from which it is received and the destination that the packet is directed to.

IR transmission

IR transmission is through compare mode. Outmode 5 (Reset) is used for transmitting a “0” and Outmode 7 Set /Reset mode is used for transmitting a “1”.When the entire packet is transmitted, the IR is put back to the receive mode. Bit time expiry interrupt is also used for transmission.

LCD Display

LCD module in the Sensor board displays the following information in cyclic mode once every 2 seconds when the device has joined the network:

1. Firmware Version (TEST FIRM)
2. Join Status (DISCOVER, SECURING, JOINING, JOINED, NOT JOINED)
3. RSSI value received from the Radio (for example,.RSSI -78)
4. RSQI value received from the Radio. (for example, RSQI 243)

Prior to the device joining the network, only the firmware version and the join status is displayed. The function “display_task()” is called every 1 second from the main loop.

This function internally checks for 2 second time-out. Join status information is received from the Radio by querying for the join status. When the device joins, the RSSI and RSQI values are received from the Radio periodically every 1 minute. The previous value is displayed until the next reception of RSSI and RSQI values. The current software version is displayed as "TESTFIRM". Each character on the LCD is an 11 segment display and is capable of displaying a maximum of 8 characters.

UDO Object

The UDO object is used for upgrading the device firmware. The new firmware is written into the EEPROM and on power can be loaded into the flash with the help of a bootloader. The current UDO implementation only accepts the firmware image and stores the image in the external EEPROM. The validation of the received firmware and loading/using the received firmware is not implemented.

REFERENCE -EXTERNAL

For the UDO functionality and the state machine, refer to ISA100 specifications document.

The function "udo_execute_ind ()" is called when the sensor receives an execute request. This function internally calls either the "start_dwnld()" , " downld_data ()" , "end_dwnld ()" functions as per the request.

"Start_dwnld()" function checks for the current state, block size, download size, and operating mode before erasing the EEPROM section to which the updated firmware has to be downloaded.

The "downld_data()" function is used for writing the data in the EEPROM. The memory location from 0x1100 to 0xFFFF is utilized for storing the data. EEPROM is written through SPI1 and is implemented through polling.

The function "end_dwnld()" sets the return code of the ESDB(execute Service descriptor block) based on the reason for end Download - complete/ client abort/ incomplete and updates the UDO variable state accordingly.

Currently, no checks are implemented for EEPROM write and the data that is downloaded through the bin file is written from EEPROM address 0x1100 onwards.

When the download is completed, a write request (which calls the function "udo_write_ind()") is issued to apply the firmware change; the Sensor then goes for a soft reset to apply the changes. The current code does not support the writing of the software into the flash.

The function "udo_get_param_info" is called following a read request and it returns the value of the UDO parameter for which the request is issued.

8. Mechanical drawing of ISA100 Radio Module

Figures 10 and 11 explains the mechanical drawing of ISA100 Radio Module.

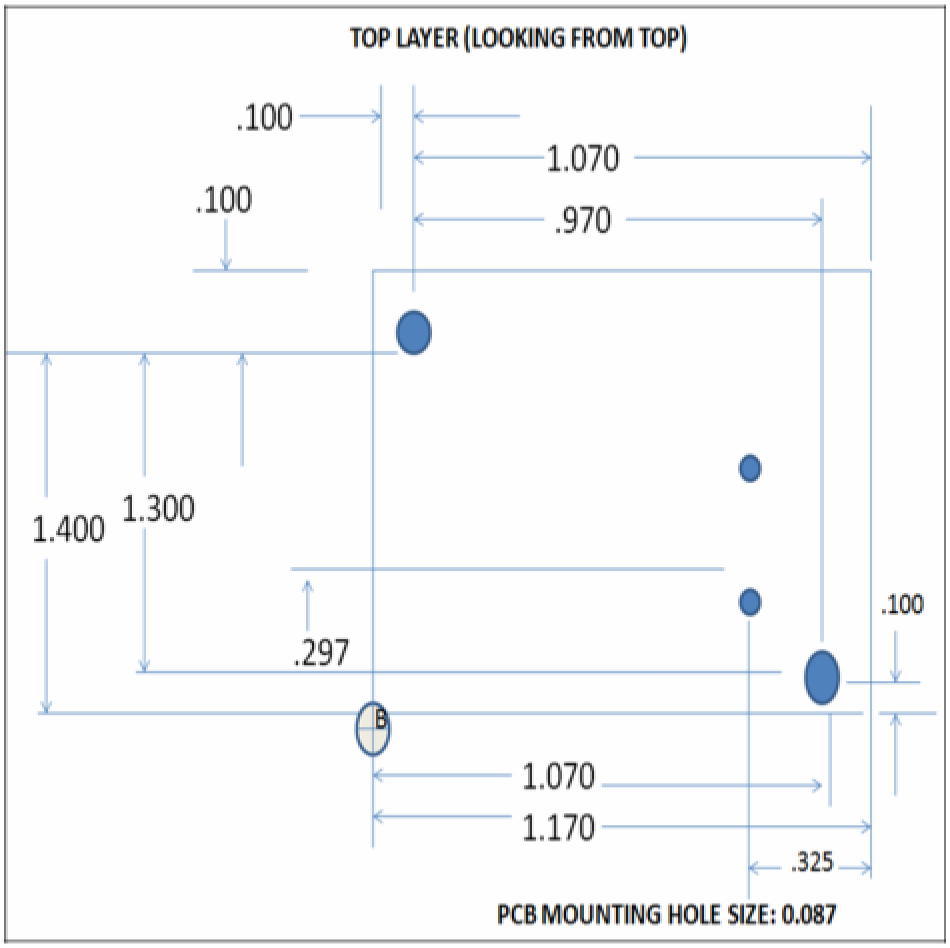


Figure 12: Mechanical Drawing of ISA 100(1)

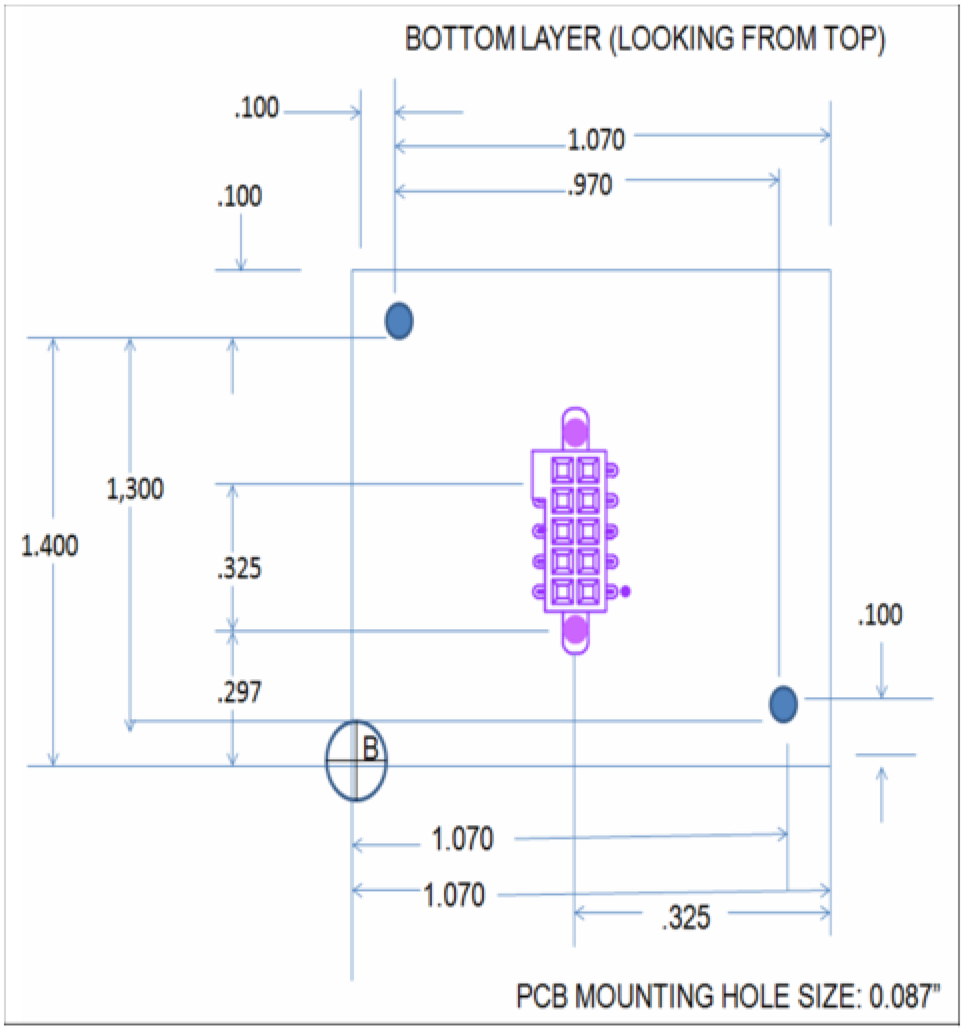

BOTTOM LAYER (LOOKING FROM TOP)
.100
1.070
.970
.100
1,300
1.400
.325
.100
.297
B
1.070
1.070
.325
PCB MOUNTING HOLE SIZE: 0.087"

Figure 13: Mechanical Drawing of ISA 100(2)

9 Power Level Settings for FCC and IC

Below table shows the power level Settings for respective antennae for FCC and IC

Sl.No	**Antenna**	**Antenna gain**	**Power level settings**
1	Centurion MAF94152 Omni Antenna	-2	16
2	L-COM HG2402RDR-RSP "Rubber-Duck"Omni Antenna	2	14
3	EM wave EM-B14503-MMP-Z6	4	14
4	L-COM HGV-2404U Omni Antenna	4	14
5	L-COM HGV-2409U Omni Antenna	8	10
6	L-COM HG-2414D remote Dish Antenna	14	6

Table 9: Antenna and power level settings FCC

Appendix A: Compliance Statements

A.1 FCC Compliance Statements

This device complies with Part 15 of the FCC rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference.
2. This device must accept any interference received including interference that may cause undesired operation of this device.

Your authority to operate equipment is void; if the changes or modifications are not expressly approved by the party responsible for compliance.

To comply with the FCC RF exposure compliance requirements, this device and its antenna must not be co-located or operated in conjunction with any other antenna or transmitter. However, this device and its antenna can be co-located or operated in conjunction with other antenna or transmitter, if installed in compliance with the FCC Multi Transmitter procedures.

To inherit the modular approval, you must install the antennas for this transmitter at a distance of 20cm from all persons and must not be co-located or operated in conjunction with any other antenna or transmitter.
To the OEM Installer:
Label the FCC ID on the final system with "Contains FCC ID : S5751454941 or "Contains transmitter module FCC ID: S5751454941."
Transmitter module must be installed and used in strict accordance with the manufacturer's instructions.
For a Class B digital device or peripheral, the instructions furnished to the user shall include the following or similar statement, placed in a prominent location in the text of the manual:
NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in an installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures: —Reorient or relocate the receiving antenna. —Increase the separation between the equipment and receiver. —Connect the equipment into an outlet on a circuit different from that to which the receiver is connected. —Consult the dealer or an experienced radio/TV technician for help
Changes or modifications to this equipment not expressly approved by Honeywell International INC may void the user's authority to operate this equipment.

TIP

The buyer of the module who will incorporate this module into his host must submit the final product to the manufacturer of the module. The manufacturer of the module WILL VERIFY that the product is incorporated in the host equipment in a way that is represented by the testing as shown in the test report.

A.2 IC Compliance Statements

Pour se conformer à l'IC, de conformité de cet appareil et son antenne ne doivent pas être situés ou exploités avec toute autre antenne ou émetteur. Toutefois, ce dispositif et son antenne ne peut être co-implantés ou exploités conjointement avec autre antenne ou un autre émetteur, s'il est installé en conformité avec les procédures de l'émetteur Multi IC.
D'hériter de l'approbation modulaire, vous devez installer les antennes de cet émetteur à une distance de 20cm à partir de toutes les personnes et ne peuvent pas être situés ou exploités avec toute autre antenne ou émetteur.
À l'équipementier de l'installateur :
l'étiquette IC sur le système final avec "Contient des IC : 573W-51454941 ou "contient le module émetteur IC : 573W-51454941."
module de l'émetteur doit être installé et utilisé conformément aux instructions du fabricant.

Ces appareils sont conformes à la partie 15 des règles de la IC. L'opération est soumise aux deux conditions suivantes:
(1) Ces dispositifs ne peuvent pas causer d'interférences nuisibles, et
(2) Ces appareils doivent accepter toute interférence reçue, y compris les interférences pouvant entraîner un fonctionnement nuisible.
Déclaration d'exposition RF
Cet équipement est conforme aux limites d'exposition aux rayonnements de la IC établies pour un environnement incontrôlé. Cet équipement doit être installé et utilisé avec une distance minimale de 20 cm entre le radiateur et votre corps.

Cet équipement a été testé et s'est avéré conforme aux limites d'un appareil numérique de classe B, conformément à la partie ICES-003 règles de la ISED.

TIP

The buyer of the module who will incorporate this module into his host must submit the final product to the manufacturer of the module. The manufacturer of the module WILL VERIFY that the product is incorporated in host equipment in a way that is represented by the testing as shown in the test report.

To comply with the IC, this unit and its antenna must not be located or operated with any other antenna or transmitter. However, this device and its antenna may not be co-located or operated in conjunction with any other antenna or transmitter, if installed in accordance with the procedures of the Multi IC transmitter.
To inherit the modular approval, you must install the antennas of this transmitter at 20cm from all people and cannot be located or operated with any other antenna or transmitter.
To the installer's equipment supplier:
the IC tag on the final system with "Contains ICs: 573W-51454941 or" contains the IC emitter module: 573W-51454941. "
transmitter module must be installed and used in accordance with the manufacturer's instructions.

These devices comply with Part 15 of the IC rules. The transaction is subject to the following two conditions:
(1) These devices may not cause harmful interference, and
(2) These devices must accept any interference received, including interference that may cause undesired operation.
RF exposure statement
This equipment complies with the IC radiation exposure limits established for an uncontrolled environment. This equipment must be installed and operated with a minimum distance of 20 cm between the radiator and your body.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to ISED Part ICES-003 Rules.

LIST OF APPROVED ANTENNAE

To know the recommended Antennae and their respective power level settings, please refer Table 9.

A.3 Software Compliance

The ISA100 field device software stack running in the Gen X Radio board is certified by the WCI (Wireless Compliance Institute) to be in compliant with 2009 version of ISA100 specification standard.

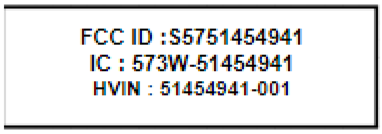
Appendix B: Agency Label Information

B.1 FCC/IC Labels

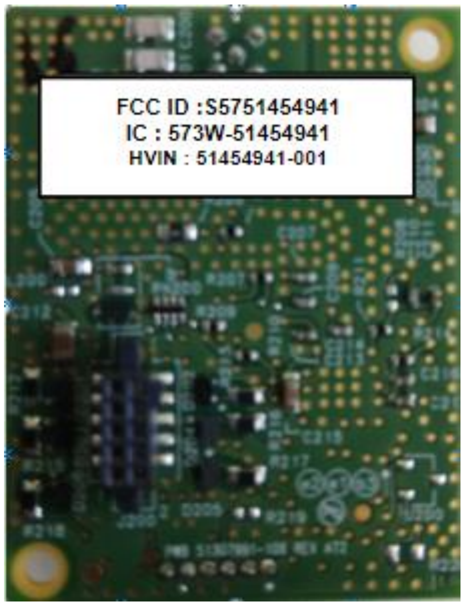
- RF MOD: 51454941
- FCC ID: S5751454941
- IC : 573W -51494541
- HVIN : 51454941 - 001

Label Location in GXRM and HOST

Label diagram:

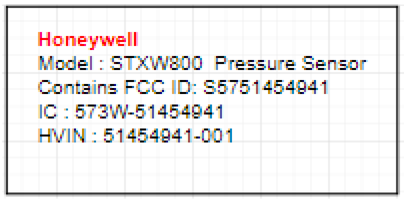

FCC ID :S5751454941
IC : 573W-51454941
HVIN : 51454941-001

Label Location:



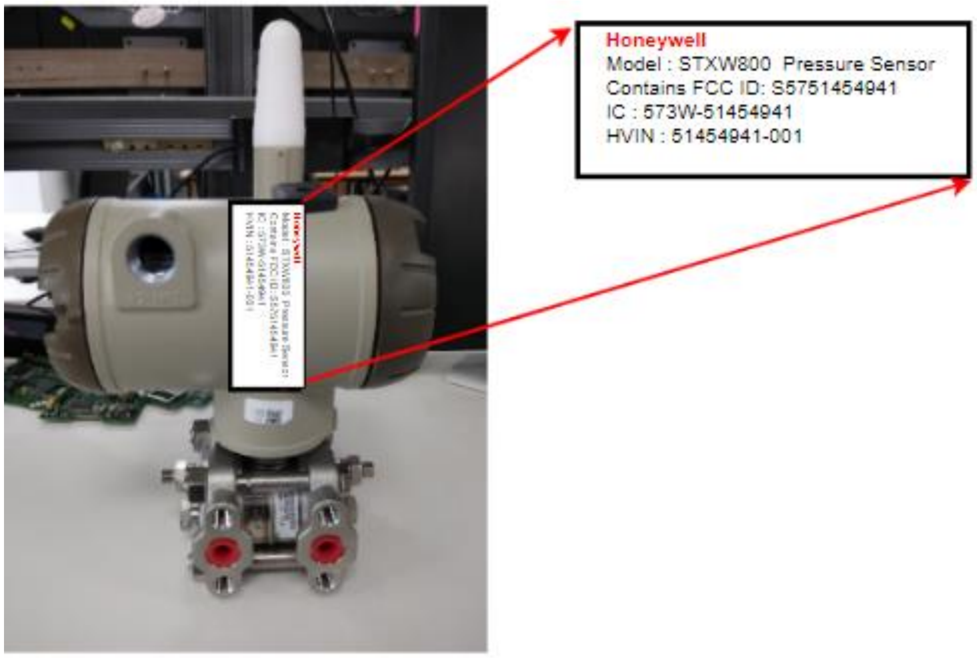
The Label is pasted on the backside of the board.

Label Diagram for Host:



Honeywell
Model : STXW800 Pressure Sensor
Contains FCC ID: S5751454941
IC : 573W-51454941
HVIN : 51454941-001

Label Location diagram:



Honeywell
Model : STXW800 Pressure Sensor
Contains FCC ID: S5751454941
IC : 573W-51454941
HVIN : 51454941-001

Appendix C: Programming GXRM

C.1 Introduction

This section gives information on the hardware and software tools that are needed as flash hex image for the Honeywell Radio board.

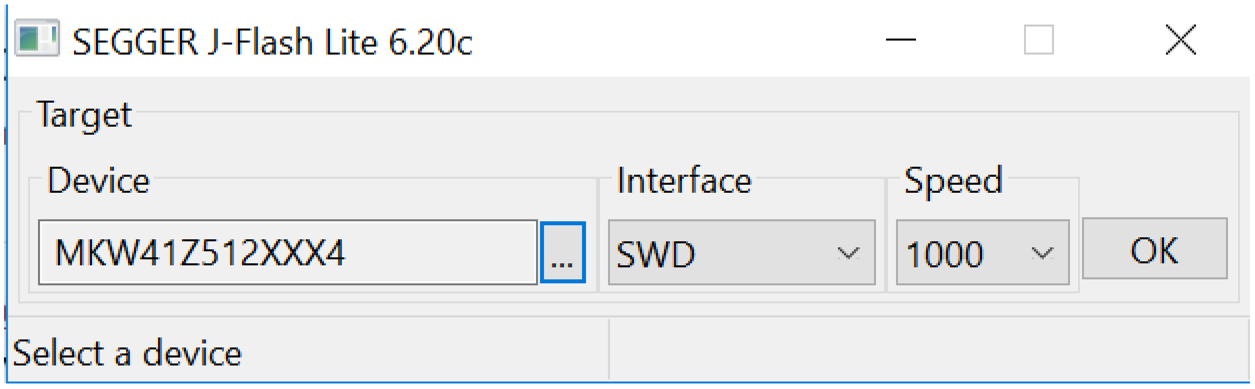
Hardware requirement
The J-Link/J-Trace Debugger is needed to flash the image.

Software requirement
The SEGGER J-flash lite tool (version 6.20) needs to be downloaded from website (https://www.segger.com/downloads/jlink/JLink_Windows_beta.exe), which is a freeware and can be used for flashing the radio board image.

C.2 Software Configuration

This section gives information on what configuration need to be used in the tool to flash the image.
After installing the software, the following steps are to be followed to configure the device.

Step 1:
Open the J-Flash lite tool, and select the mentioned options as in the image below



Device: MKW41Z512XXX4
Interface: SWD
Speed: 1000
Select a device

Figure 14:J-Flash tool device selection

Step 2:
After configuring the device select the HEX file that must be flashed, in the data file section

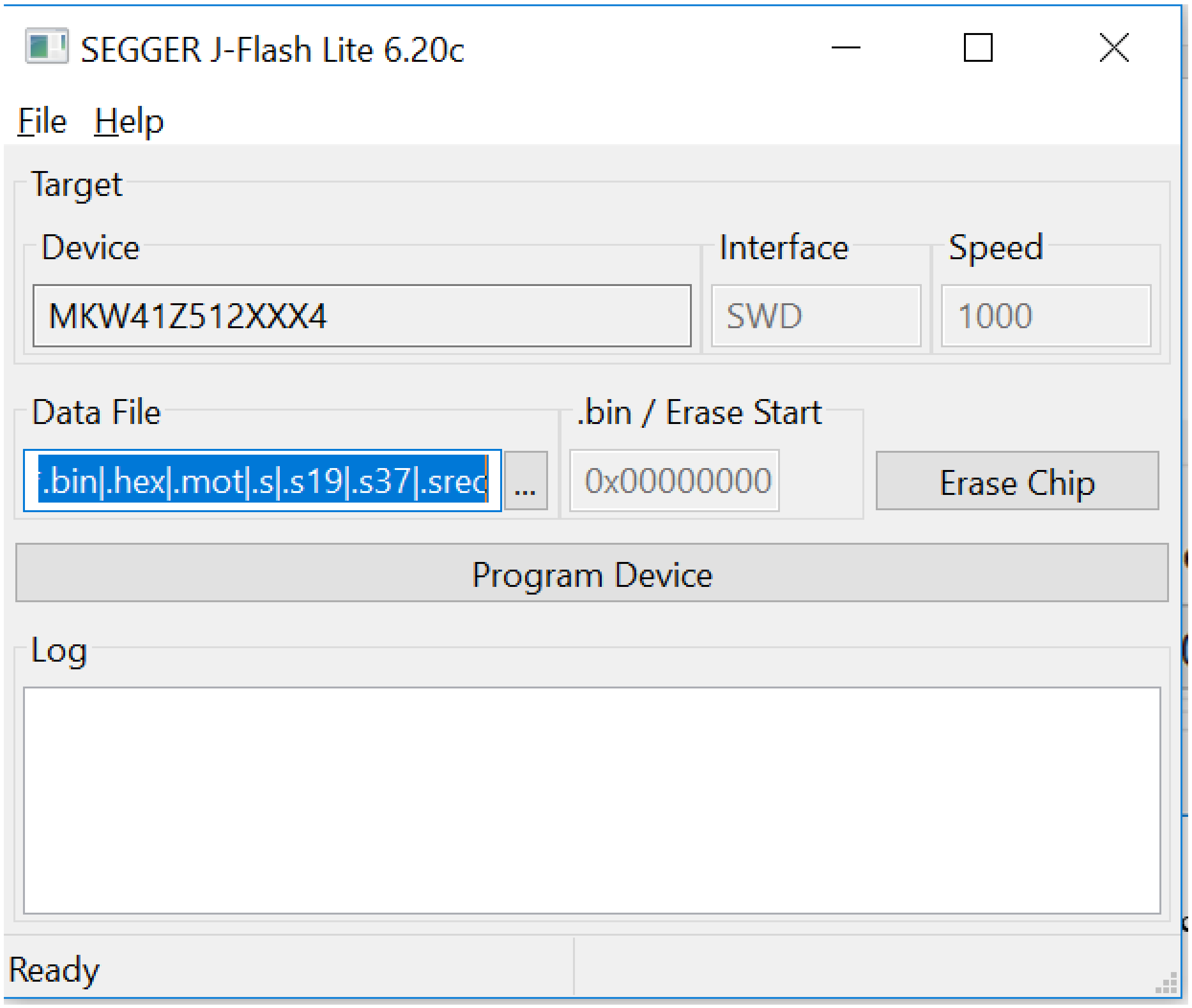

SEGGER J-Flash Lite 6.20c

Figure 15:J-Flash tool data file selection

Step 3:
After selecting the HEX file, click on Program Device option. The program will be flashed to the device.

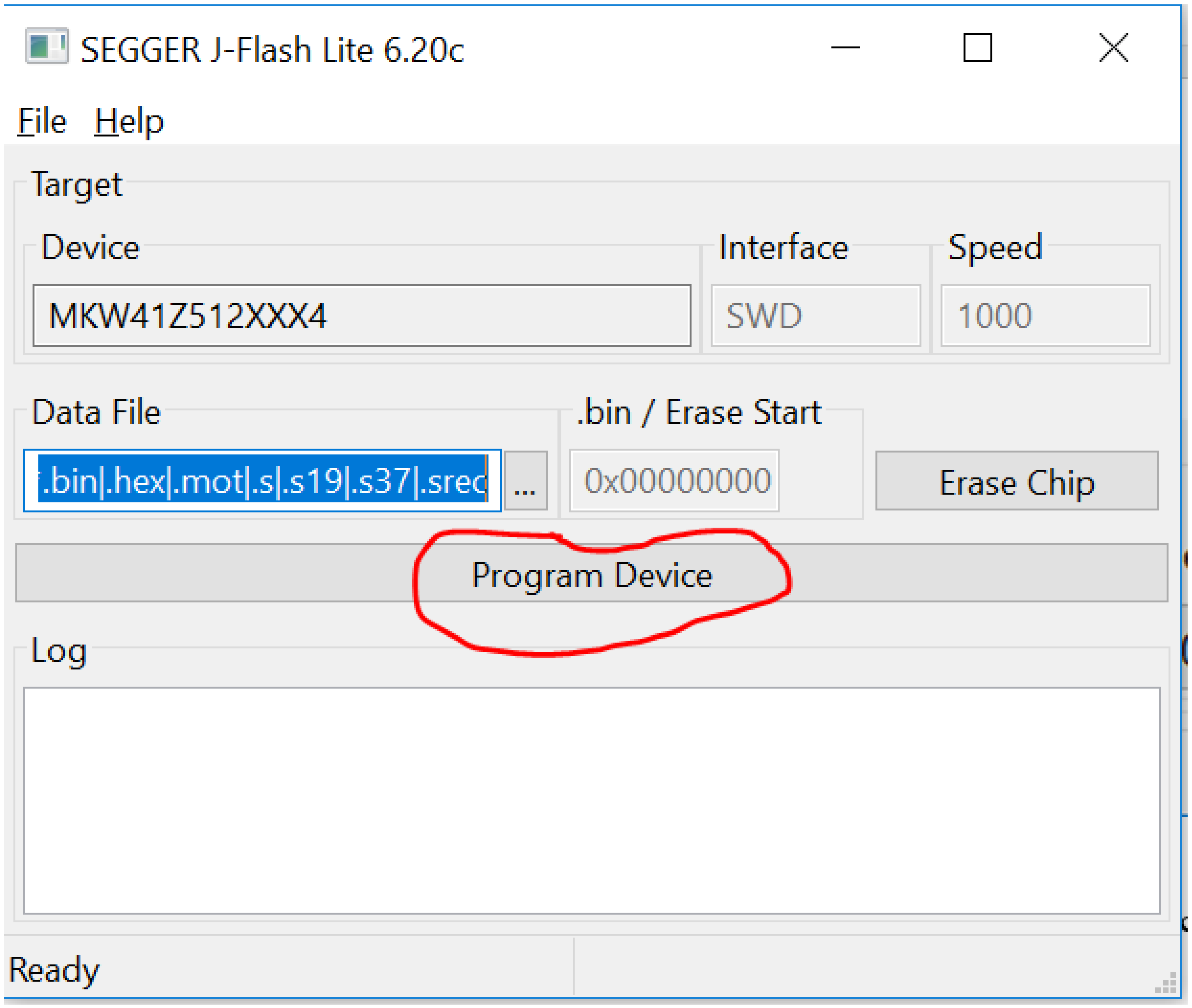


Figure 16: J-Flash tool device programming

Once the device has been flashed with the firmware the "Programming done" will be displayed in the Log section.