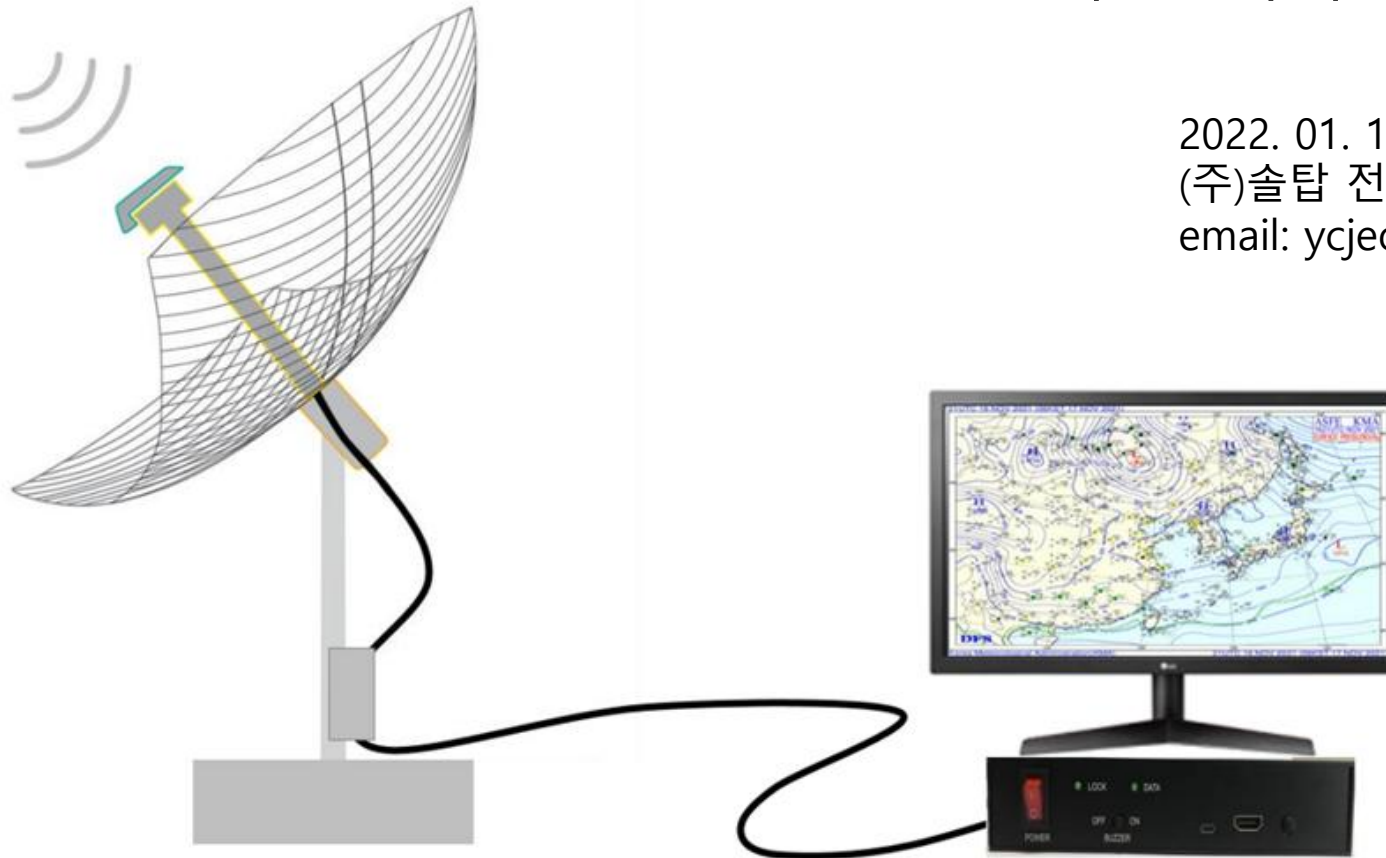


## GK2A LRIT 수신 처리 과정

2022. 01. 17

(주)솔탑 전영찬 책임 연구원

email: ycjeon@soletop.com



주의) 본 문서의 내용과 제공된 라이브러리에 대해 제 3자에게 배포를 금지합니다.

- 방송 자료 소개
- 수신 장비 개요
- 수신 처리 개요
- 수신 처리 흐름
- 단위 기술 설명
- 실습 시 제공 자료
- 샘플 소스 코드

- ❖ 어떤 자료를 방송하는가?
- ❖ FD 방송 자료의 형태는?
- ❖ FD외 방송 자료의 형태는?
- ❖ 방송 파일 형태
- ❖ 방송 자료의 파일 정보

어떤 자료를 방송하는가?

- 방송 기관: 국가기상위성센터(진천 소재)
- 수신 대상: 공개 방송(장비를 갖춘 사용자 누구나 무료)
- 수신 가능 지역: GK2A LRIT 커버리지
- 방송 시간표: 별도 제공

<방송 시간표 예시>

TIME (UTC)	ABBR_ID	AREA	DISSEMINA
000940-001000	EGMSG001		○
001006-001236	FD001	FD	○
001940-002000	EGMSG002		○
002006-002236	FD002	FD	○
002940-003000	EGMSG003		○
003006-003236	FD003	FD	○
003940-004000	EGMSG004		○
004006-004236	FD004	FD	○
004940-005000	EGMSG005		○
005006-005236	FD005	FD	○
005940-010000	EGMSG006		○
010006-010236	FD006	FD	○
010940-011000	EGMSG007		○
011006-011236	FD007	FD	○
011300-011330	TYIA001		○
011940-012000	EGMSG008		○
012006-012236	FD008	FD	○
012940-013000	EGMSG009		○
013006-013236	FD009	FD	○
013940-014000	EGMSG010		○

<방송 자료 종류>

자료 약어(ABBR_ID)	자료 내용	방송 주기
ANT	다음날 방송 스케줄	1일 1회
COMSFOG	안개	1일 4회
COMSIR1	적외105(천리안포맷)	(생략)
EGMSG	긴급 메시지	발생 시
GWW3F	전구파랑 예상도	(생략)
<b>FD</b>	<b>적외105</b> (LRIT에는 적외105 단일 자료로 구성)	<b>10분 간격</b>
RWW3A	파랑실황도	(생략)
RWW3F	파랑실황도	(생략)
SICEA	해빙	(생략)
SSTA	해수면온도	(생략)
SSTF24	해수면온도	(생략)
SSTF48	해수면온도	(생략)
TYIA001	태풍정보	(생략)
(이하생략)	(이하생략)	

FD 방송 자료의 형태는?

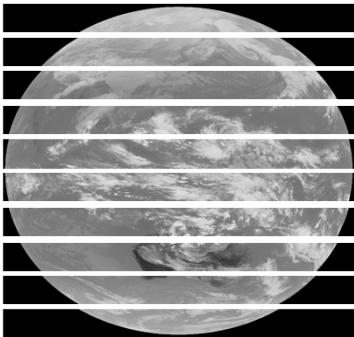
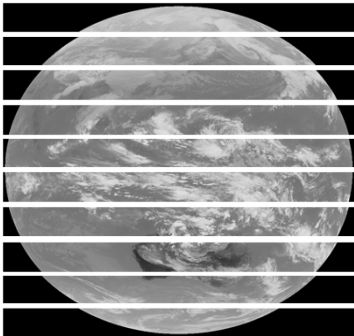
TIME (UTC)	ABBR_ID	AREA	DISSEMINATION
000940-001000	EGMSG001		O
001006-001236	FD001	FD	O
001940-002000	EGMSG002		O
002006-002236	FD002	FD	O
002940-003000	EGMSG003		O
003006-003236	FD003	FD	O
003940-004000	EGMSG004		O
004006-004236	FD004	FD	O
004940-005000	EGMSG005		O
005006-005236	FD005	FD	O
005940-010000	EGMSG006		O
010006-010236	FD006	FD	O
010940-011000	EGMSG007		O
011006-011236	FD007	FD	O
011300-011330	TYIA001		O
011940-012000	EGMSG008		O
012006-012236	FD008	FD	O
012940-013000	EGMSG009		O
013006-013236	FD009	FD	O
013940-014000	EGMSG010		O

00시 10분 06초 – 00시 12분 36초까지

- IMG\_FD\_001\_IR105\_20211215\_001006\_01.Irit
- IMG\_FD\_001\_IR105\_20211215\_001006\_02.Irit
- IMG\_FD\_001\_IR105\_20211215\_001006\_03.Irit
- IMG\_FD\_001\_IR105\_20211215\_001006\_04.Irit
- IMG\_FD\_001\_IR105\_20211215\_001006\_05.Irit
- IMG\_FD\_001\_IR105\_20211215\_001006\_06.Irit
- IMG\_FD\_001\_IR105\_20211215\_001006\_07.Irit
- IMG\_FD\_001\_IR105\_20211215\_001006\_08.Irit
- IMG\_FD\_001\_IR105\_20211215\_001006\_09.Irit
- IMG\_FD\_001\_IR105\_20211215\_001006\_10.Irit

00시 20분 06초 – 00시 22분 36초까지

- IMG\_FD\_002\_IR105\_20211215\_002006\_01
- IMG\_FD\_002\_IR105\_20211215\_002006\_02
- IMG\_FD\_002\_IR105\_20211215\_002006\_03
- IMG\_FD\_002\_IR105\_20211215\_002006\_04
- IMG\_FD\_002\_IR105\_20211215\_002006\_05
- IMG\_FD\_002\_IR105\_20211215\_002006\_06
- IMG\_FD\_002\_IR105\_20211215\_002006\_07
- IMG\_FD\_002\_IR105\_20211215\_002006\_08
- IMG\_FD\_002\_IR105\_20211215\_002006\_09
- IMG\_FD\_002\_IR105\_20211215\_002006\_10



⋮

## FD외 방송 자료의 형태는?

6 페이지

013006-013236	FD009	FD
013940-014000	EGMSG010	
014006-014236	FD010	FD
014300-014330	RWW3A001	
014940-015000	EGMSG011	
015006-015236	FD011	FD
015300-015330	TYIB001	
015940-020000	EGMSG012	
020006-020236	FD012	FD
020300-020340	SUFA03001	
020940-021000	EGMSG013	
021006-021236	FD013	FD
021300-021330	UP50A001	
021940-022000	EGMSG014	
022006-022236	FD014	FD

01시 43분 00초 – 01시 43분 30초까지

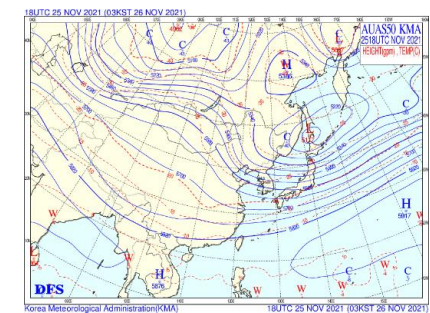
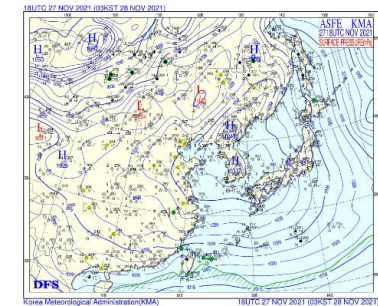
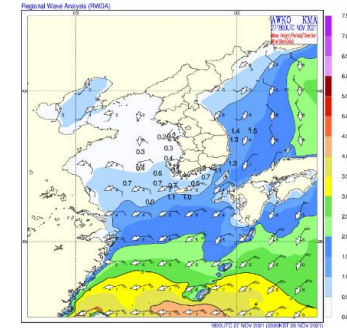
ADD\_RWW3A\_001\_20211215\_014300\_00.lrit

02시 30분 00초 – 02시 03분 40초까지

ADD\_SUFA03\_001\_20211215\_020300\_00.lrit

02시 13분 00초 – 02시 13분 30초까지

ADD\_UP50A\_001\_20211215\_021300\_00.lrit



- IMG(=FD), ADD의 두 형태만 존재. 파일 타입 별로 처리 방법이 다르기 때문에 구분해서 이해하여야 함.

파일 타입	구분 방법	자료 구조	비고
IMG(=FD)	IMG_로 시작	바이너리 데이터와 바이너리 데이터에 대한 메타 정보	GK2A AMI 관측자료
ADD	ADD_로 시작	이미지 파일, 텍스트, 또는 음성 파일(향후)	수집 자료

- 수신 파일명 사례

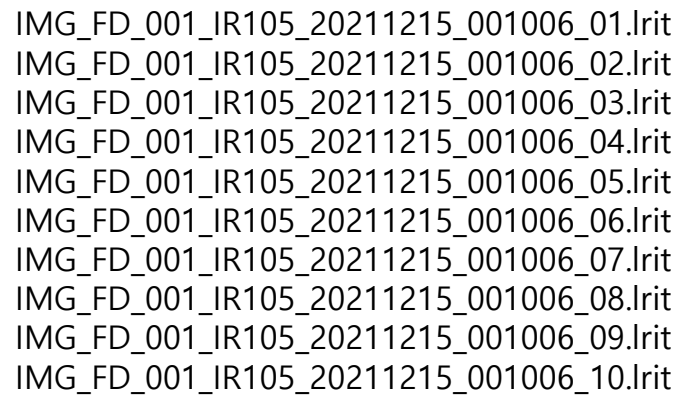
```

907 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_023_IR105_20211215_035006_07.lrit"
908 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_023_IR105_20211215_035006_08.lrit"
909 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_023_IR105_20211215_035006_09.lrit"
910 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_023_IR105_20211215_035006_10.lrit"
911 File Received: "/home/receiverpi/gk-2a/data/received/ADD_GWW3F_001_20211215_035300_00.lrit"
912 File Received: "/home/receiverpi/gk-2a/data/received/ADD_GWW3F_002_20211215_035345_00.lrit"
913 File Received: "/home/receiverpi/gk-2a/data/received/ADD_GWW3F_003_20211215_035430_00.lrit"
914 File Received: "/home/receiverpi/gk-2a/data/received/ADD_GWW3F_005_20211215_035600_00.lrit"
915 File Received: "/home/receiverpi/gk-2a/data/received/ADD_GWW3F_006_20211215_035645_00.lrit"
916 File Received: "/home/receiverpi/gk-2a/data/received/ADD_GWW3F_007_20211215_035730_00.lrit"
917 File Received: "/home/receiverpi/gk-2a/data/received/ADD_GWW3F_008_20211215_035815_00.lrit"
918 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_024_IR105_20211215_040006_01.lrit"
919 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_024_IR105_20211215_040006_02.lrit"
920 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_024_IR105_20211215_040006_03.lrit"
921 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_024_IR105_20211215_040006_05.lrit"
922 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_024_IR105_20211215_040006_06.lrit"
923 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_024_IR105_20211215_040006_07.lrit"
924 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_024_IR105_20211215_040006_08.lrit"
925 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_024_IR105_20211215_040006_09.lrit"
926 File Received: "/home/receiverpi/gk-2a/data/received/IMG_FD_024_IR105_20211215_040006_10.lrit"
927 File Received: "/home/receiverpi/gk-2a/data/received/ADD_GWW3F_009_20211215_040300_00.lrit"
928 File Received: "/home/receiverpi/gk-2a/data/received/ADD_GWW3F_010_20211215_040345_00.lrit"
929 File Received: "/home/receiverpi/gk-2a/data/received/ADD_GWW3F_011_20211215_040430_00.lrit"

```

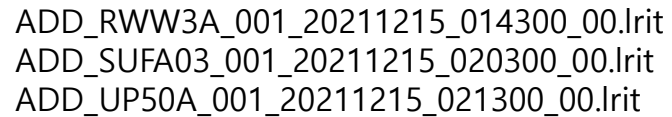
- 파일명

- IMG\_지역\_당일방송순번\_관측채널\_방송날짜\_방송시작시각\_세그먼트순번.lrit



IMG\_FD\_001\_IR105\_20211215\_001006\_01.lrit  
IMG\_FD\_001\_IR105\_20211215\_001006\_02.lrit  
IMG\_FD\_001\_IR105\_20211215\_001006\_03.lrit  
IMG\_FD\_001\_IR105\_20211215\_001006\_04.lrit  
IMG\_FD\_001\_IR105\_20211215\_001006\_05.lrit  
IMG\_FD\_001\_IR105\_20211215\_001006\_06.lrit  
IMG\_FD\_001\_IR105\_20211215\_001006\_07.lrit  
IMG\_FD\_001\_IR105\_20211215\_001006\_08.lrit  
IMG\_FD\_001\_IR105\_20211215\_001006\_09.lrit  
IMG\_FD\_001\_IR105\_20211215\_001006\_10.lrit

- ADD\_약어\_당일방송순번\_방송날짜\_방송시작시각\_00.lrit

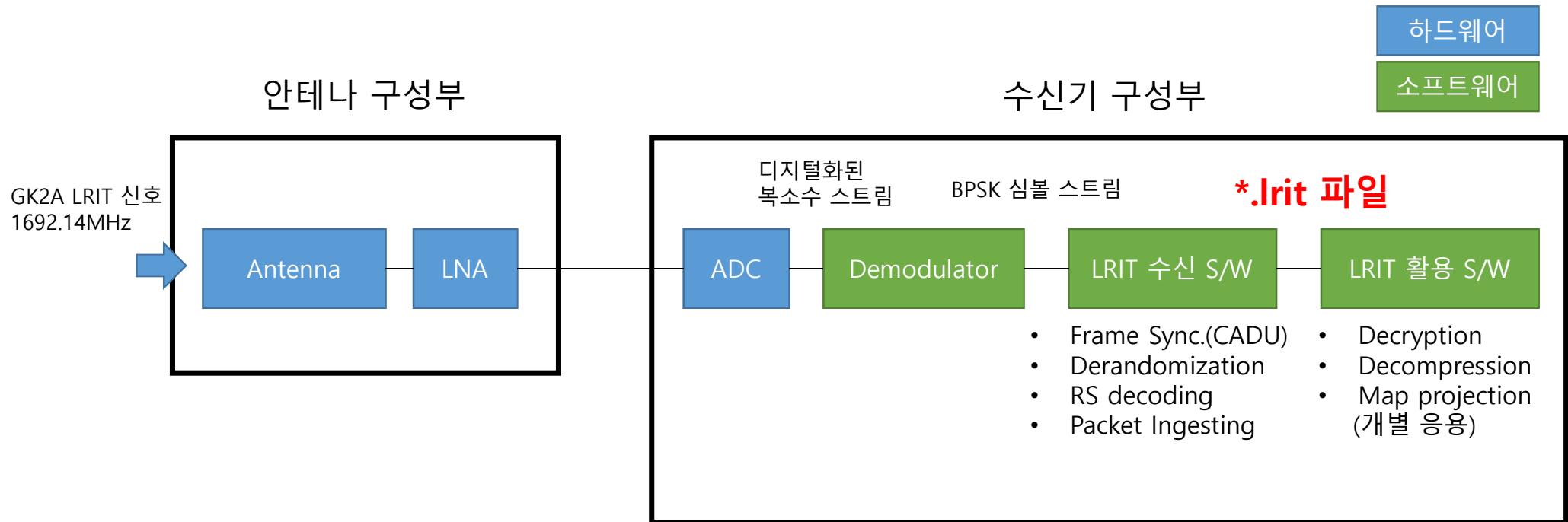


ADD\_RWW3A\_001\_20211215\_014300\_00.lrit  
ADD\_SUFA03\_001\_20211215\_020300\_00.lrit  
ADD\_UP50A\_001\_20211215\_021300\_00.lrit

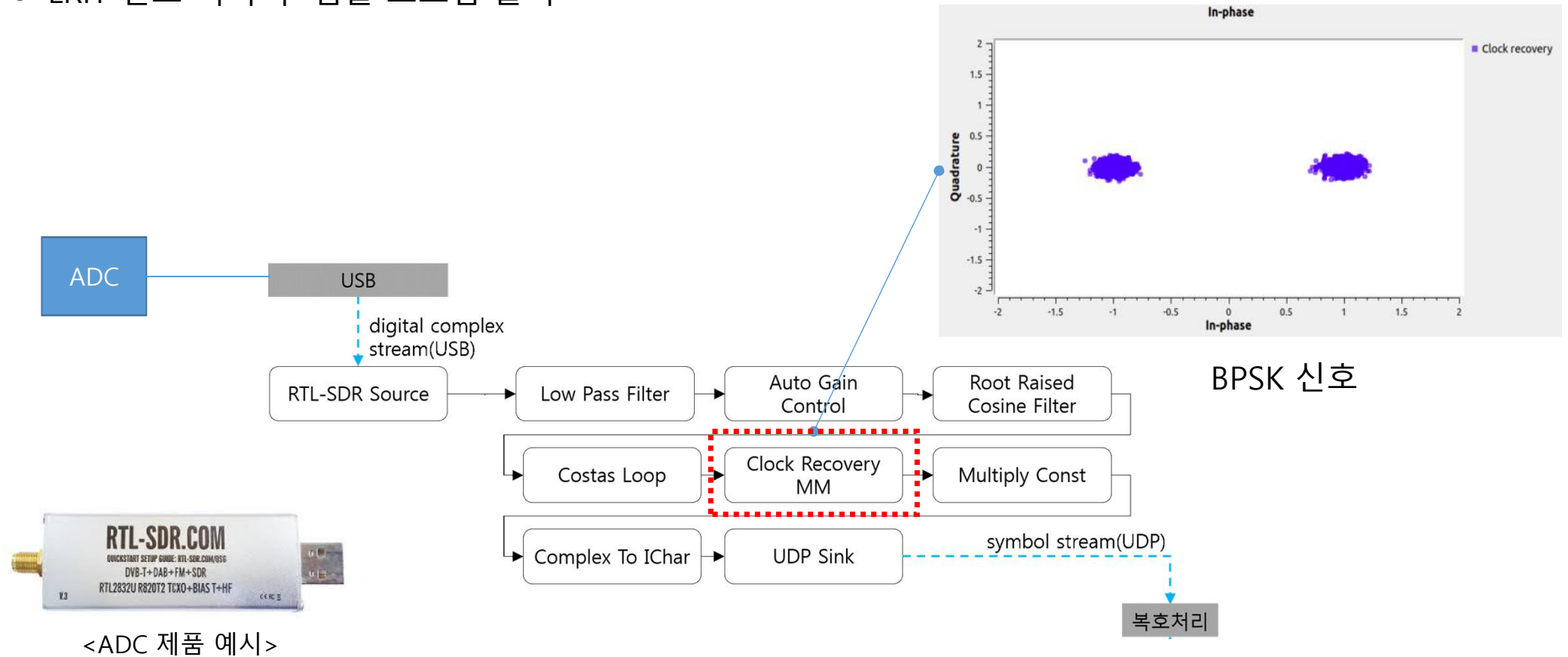


1. GK2A LRIT 신호 수신 장비 구성과 역할
2. 디모듈레이터 소개

- LRIT 신호 수신 장비 및 역할
- 수신 소프트웨어 구성 및 역할



- LRIT 신호 처리 후 심볼 스트림 출력



1. GK2A LRIT 수신을 위한 규격 문서 및 참고 문서
2. 실습 범위

- GK2A LRIT Mission Specification(2019년 발행)
  - GK2A LRIT 임무 규격(설명이 부족함)
- COMS LRIT Mission Specification(2010년 발행)
  - COMS LRIT 임무 규격 문서(설명이 잘 되어 있음)
- CGMS LRIT/HRIT global specification(1993년 발행)
  - LRIT/HRIT Global Spec. 문서(LRIT/HRIT 국제 규격 문서)

---

COMS: 천리안 1호(2010년 발사~2019년, 임무 종료)

GK2A: 천리안 2A호 (2018년 발사~ 10년간 운영)

CGMS: Coordination Group for Meteorological Satellites(기상위성협력그룹)

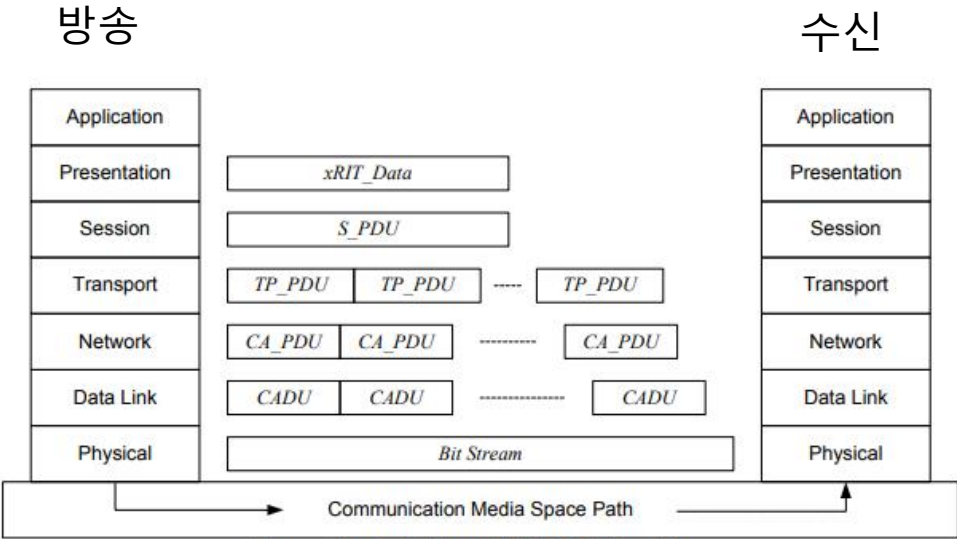


Figure 2.1 Definition of GK2A LRIT Data Type

Table 2.1 OSI Layer Functionalities for GK2A LRIT Service

OSI 7 layers	Layer functionalities
Application layer	Acquisition of application data
Presentation layer	Image segmentation, LRIT file structuring
Session layer	Compression (if required) Encryption (if required)
Transport layer	Determination of APID Split of files into source packet
Network layer	Determination of VCID
Data link layer	Multiplexing, Error of block unit detection, Reed-Solomon encoding Randomization Attachment of sync marker
Physical layer	Serialization, Viterbi encoding, Modulation

실습 범위

1. BPSK Symbol Stream to Frame Sync, Viterbi Decoding 소개
2. CADU to VCDU 처리 과정
3. VCDU 에 대한 이해
4. 각 가상채널 별 VCDU를 Source Packet으로 조립
5. Source Packet에서 TP\_File 조립
6. TP\_File의 구조 및 LRIT 파일로 변환
7. LRIT 파일의 헤더 구조 확인
8. Image File Type의 LRIT 파일의 헤더 구조
9. Image File Type의 LRIT 파일 암호 해제
10. Image File Type의 LRIT 파일 압축 해제
11. LRIT 파일의 파일명 필드 읽기
12. Image File Type의 Segmentation된 자료(Presentation Layer)를 병합하여  
Application Layer 수준의 자료로 복원
13. Additional File Type의 LRIT 파일 처리

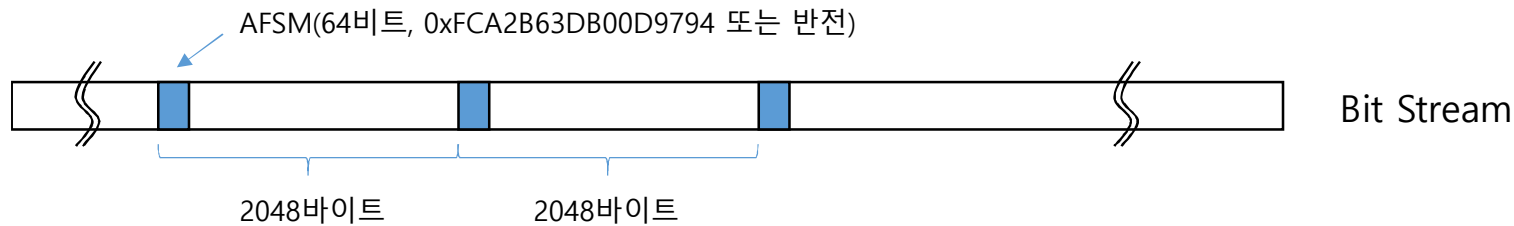
## BPSK Symbol Stream to Frame Sync, Viterbi Decoding 소개

16 페이지

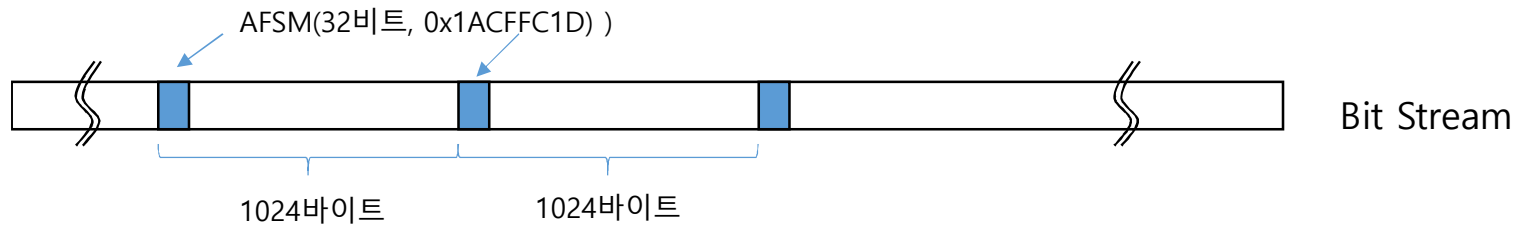
- Symbol: -128~127 범위의 값을 갖는 x, y 쌍
- 미리 알고 있는(정의된 값) AFSM(Attached Frame Sync. Marker) 값을 스트림에서 찾을
- ASFM 형태( $\frac{1}{2}$  Viterbi encoded)

0xFCA2B63DB00D9794	phase 0	11	11	11	00	10	10	00	10	10	11	01	10	00	11	11	01	10	11	00	00	00	00	11	01	10	01	01	11	10	01	01	00
0x56FBD394DAA4C1C2	phase 90	01	01	01	10	11	11	10	11	11	01	00	11	10	01	01	00	11	01	10	10	10	10	01	00	11	00	00	01	11	00	00	10
0x035D49C24FF2686B	phase 180	00	00	00	11	01	01	11	01	01	00	10	01	11	00	00	10	01	00	11	11	11	11	00	10	01	10	10	00	01	10	10	11
0xA9042C6B255B3E3D	phase 270	10	10	10	01	00	00	01	00	00	10	11	00	01	10	10	11	00	10	01	01	01	01	10	11	00	11	11	10	00	11	11	01

### ● 입력 자료

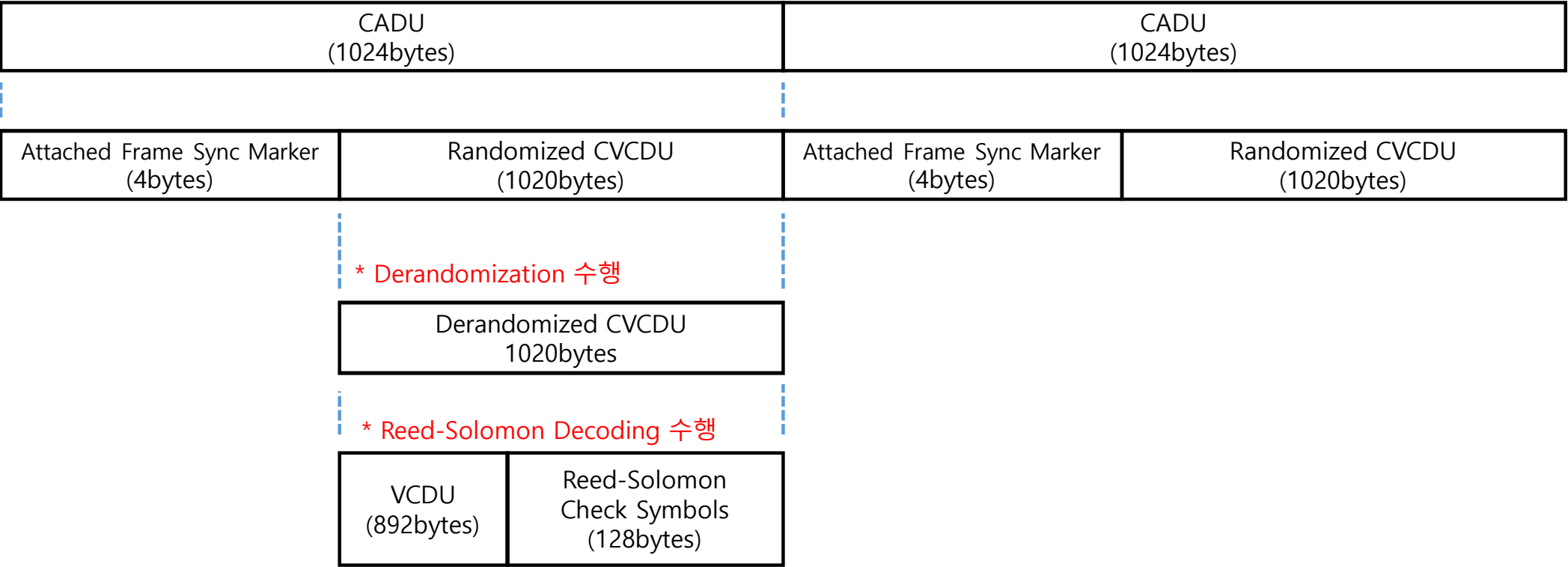


### ● 출력 자료

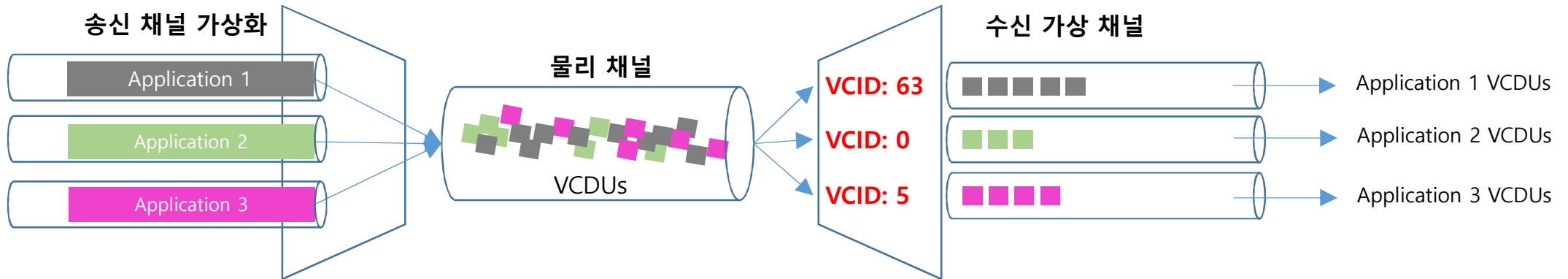




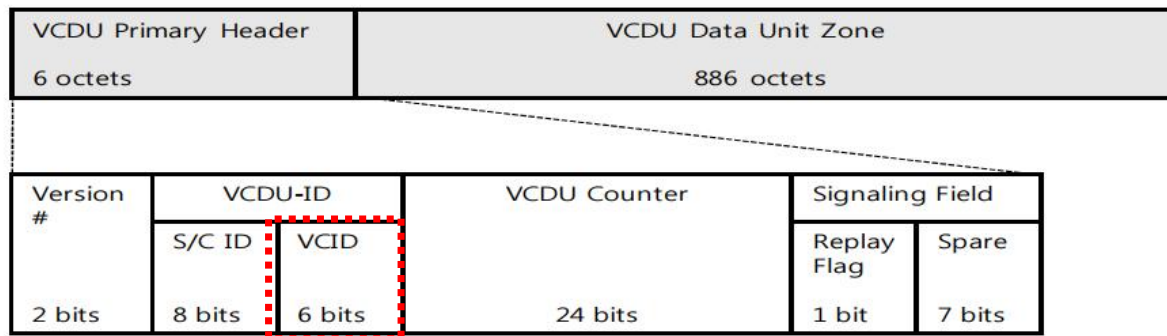
- 각 CADU 단위로 데이터 구간에 대해 처리



- 하나의 물리 채널을 통해 여러 Application 자료를 송수신하기 위한 자료 단위



- VCDU 자료 구조

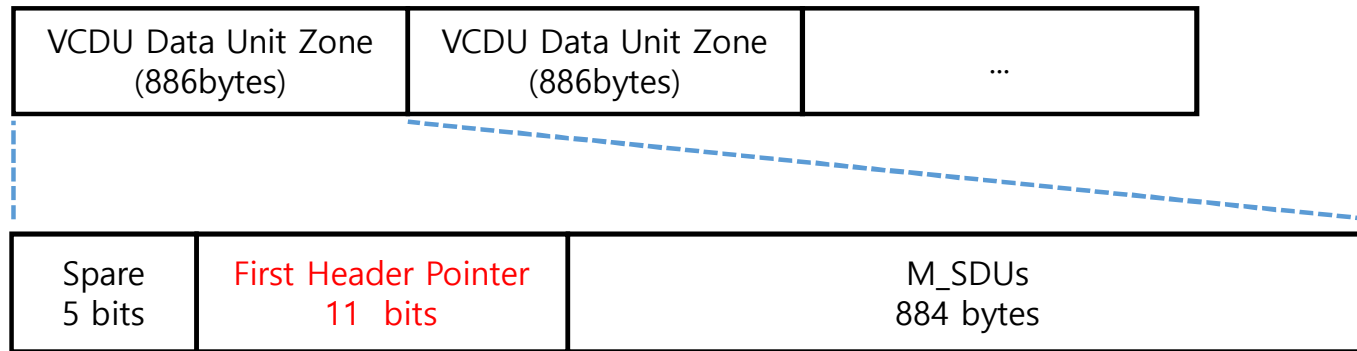


## 각 가상채널 별 VCDU를 Source Packet으로 조립(1/2)

19 페이지

- First Header Pointer 값을 읽어서 M\_SDUs 처리

가상채널 A번:



## 각 가상채널 별 VCDU를 Source Packet으로 조립(2/2)

20 페이지

- First Header Pointer(11bits):
  - 0x07FF 이면 계속 되는 Source Packet에 속하는 M\_SDUs
  - 0x07FF 가 아니면 이전의 소스패킷과 다음 소스패킷을 분리하는 offset

가상채널 A번:

M_SDUs 884 bytes	M_SDUs 884 bytes	...	M_SDUs 884 bytes	M_SDUs 884 bytes	...
---------------------	---------------------	-----	---------------------	---------------------	-----

First Header Pointer:

0	0x07FF	0x02A8 (예시)	0x07FF	0x07FF	
---	--------	----------------	--------	--------	--



- Data Field 블록을 이어 붙이면 TP\_File

Table 6.3 Source Packet Structure

Source Packet Header							Packet Data Field	
Packet Identification				Packet Sequence Control		Packet Length	Data Field	
Version	Type	Secondary Header Flag	APID	Sequence Flag	Packet Sequence Count		Application Data Field	CRC
3 bits	1 bit	1 bit	11 bit	2 bits	14 bits	16 bits	Var.	16 bits
2 bytes				2 bytes		2 bytes	Max. 8190 bytes	2 bytes

가상채널 A번:

Source Packet #1 Data Field	Source Packet #2 Data Field	Source Packet #3 Data Field	...	Source Packet #N Data Field
--------------------------------	--------------------------------	--------------------------------	-----	--------------------------------

Sequence Flag

01

00

00

00

10

TP\_File



주의) Packet Length의 값은 Data Field 길이보다 1이 크게 되어 있음.  
즉, Packet Length - 1 을 하여야 Data Field의 길이와 동일함.

- 현재 단계에서 S\_PDU + Filler 영역을 저장하면 LRIT 파일(압축, 암호화된) 이 됨

가상채널 A번:

TP_Header		S_PDU	Filler
File Counter 16 bits	File Length 64 bits	1 ~ (2 <sup>64</sup> -1) bits	0~7 bits

LRIT File

## LRIT 파일의 헤더 구조 확인

- VCID에 따라 LRIT 파일에서 출현하는 헤더가 다름

### Appendix B: GK2A LRIT/HRIT/UHRIT APID and VCID

In the future, the actual APIDs and VCIDs will be determined by NMSC's broadcasting policy. Next table shows current values of them. The APIDs and VCIDs will be determined w.r.t broadcasting data categories, not w.r.t broadcasting channels(LRIT/HRIT/UHRIT).

Category 1	Category 2	Category 3	APID	VCID
Image Data	FD	VI004	0	0
		VI005	1	
		...	...	
		IR113	14	
		IR133	15	
Additional Data	Reserved	-	32 ~ 127	1 ~ 3
	Alpha-numeric Text	-	128	4
	Additional Data	-	160	5

Table 4.16 File Type vs. Header Implementation

File types		Header record types												
		0	1	2	3	4	5	6	7	128	129	130	131	132
0	Image data file	●	●	◎	◎	◎	◎		◎	◎		◎	◎	◎
1	GTS Message													
2	Alphanumeric text file	●				◎	◎		◎					
3	Encryption key message	●				◎	◎		◎					

● As requested by [RD7] ◎ KMA mandatory use ○ KMA optional use

0 Primary header  
1 Image structure  
2 Image navigation  
3 Image data function  
4 Annotation  
5 Time stamp  
6 Ancillary text  
7 Key header

128 Image segment identification  
129 Encryption Key message header  
130 Image compensation info. header  
131 Image observation time header  
132 Image quality information header

- 참고

※ Additional Data의 Header Implementation 정보가 없는데 Alphanumeric text file 의 헤더 구조와 동일

## Image File Type의 LRIT 파일의 헤더 구조

- IMG 파일의 헤더 예시처럼 LRIT 파일에서 출현하는 헤더가 다름
- 각 헤더의 자료 구조는 Mission Spec 문서에 정의

File types		Header record types												
		0	1	2	3	4	5	6	7	128	129	130	131	132
0	Image data file	●	●	⊙	⊙	⊙	⊙		⊙	⊙		○	⊙	○

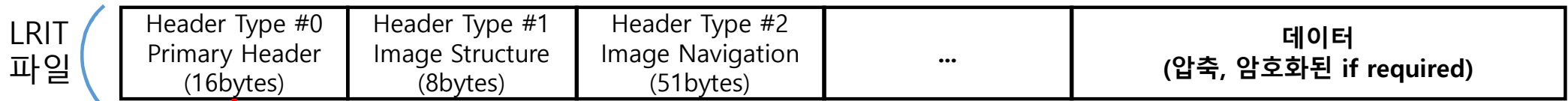


Table 4.3 Header Type #0 – Primary Header

Classification	Data Type	Data Size (Bytes)	Value	Remark
Header Type	unsigned integer	1	0	Fixed value
Header Record Length	unsigned integer	2	16	Fixed value
File Type Code	unsigned integer	1	Variable	0: Image data file 1: GTS message(Not used) 2: Alphanumeric text file 3: Encryption key message(Not used)
Total Header Length	unsigned integer	4	Variable	Total Header Record size(Bytes)
Data Field Length	unsigned integer	8	Variable	Data Field size(bits)

- Header Values는 Big Endian 형식으로 저장되어 있으므로 Little Endian 변환하여 읽어야 함



- 암호 해제를 위해 8바이트 길이의 key가 필요
- 암호 해제 정보는 Header Type #7에 정의
- 암호 해제 키 테이블에서 Key Number에 해당하는 Key Value를 테이블에서 찾아서 암호 해제에 사용
- LRIT 파일에서 Data 영역만 암호화 되어 있음

### 4.4.8 Header Type #7 – Key Header

This header provides the number of used encryption key.

Table 4.9 Header Type #7 – Key Header

Classification	Data Type	Data size (Bytes)	Value	Remark
Header Type	Unsigned integer	1	7	Fixed value
Header Record Length	Unsigned integer	2	7	Fixed value
Key Number	Unsigned integer	4	Variable	Index of the used encryption key 0: Encryption is not applied

<기상청으로부터 발급받은 암호 해제 키 테이블 구조>

Key Number	Key Value(8bytes)
0x65	0xAABBCCDDAABBCCDD
0x66	0x1122334455667788
...	...
0x70	...
0x71	...
..	...
..	...

## Image File Type의 LRIT 파일 압축 해제

- 암호 해제를 먼저 수행해야 함.
- 압축 정보는 Header Type #1에 정의

### 4.4.2 Header Type #1 – Image Structure

This header provides number of bits per pixel, number of columns, number of lines of image structure, and compression flag.

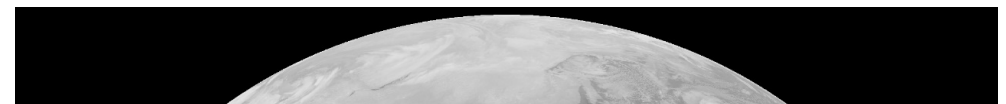
Table 4.4 Header Type #1 – Image Structure

Classification	Data Type	Data size (Bytes)	Value	Remark
Header Type	unsigned integer	1	1	Fixed value
Header Record Length	unsigned integer	2	9	Fixed value
Number of bit per pixel	unsigned integer	1	Variable	Input valid bit according to channel
Number of columns	unsigned integer	2	Variable	Variable size according to observation area and channel
Number of lines	unsigned integer	2	Variable	Variable size according to observation area and channel
Compression Flag	unsigned integer	1	Variable	Compression method 0: No compression 1: Lossless compression 2: Lossy compression

LRIT 파일은 Jpeg Lossy(손실압축) 형태로 압축됨.  
이 압축 형식은 운영체제에서 기본적으로 지원하므로 일반적인 Jpeg 파일과 동일함.

Headers ...	데이터
----------------	-----

데이터 영역을\*.jpg 파일로 저장하면  
일반적인 운영체제(Windows, ... )에서 읽을 수 있는 jpg 파일



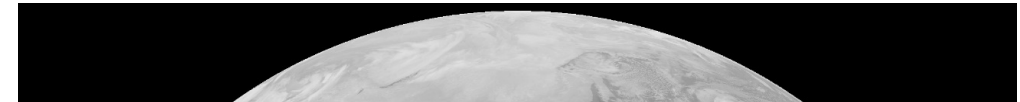
- 파일명 정보는 Header Type #4에 정의

### 4.4.5 Header Type #4 – Annotation Text

This header provides the annotation record to allow quicker and easier detection of file contents.

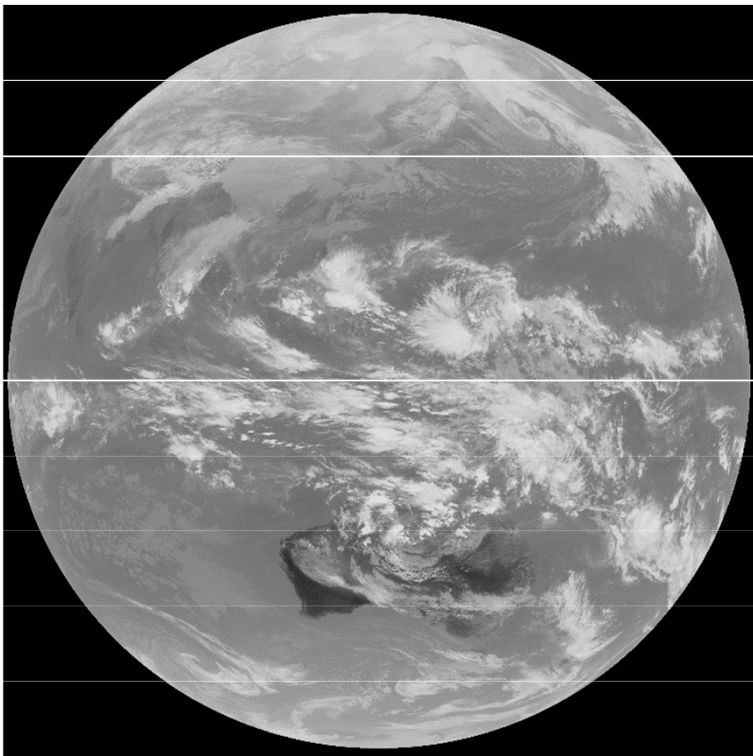
Table 4.7 Header Type #4 - Annotation

Classification	Data Type	Data size (Bytes)	Value	Remark
Header Type	Unsigned integer	1	4	Fixed value
Header Record Length	Unsigned integer	2	Variable	Max. 67
Annotation Text	Character	Variable	Variable	Max. 64 File Name IMG_FD_143_VI006_20180627_030000_01.lrit ADD_ANT_143_20180627_030000_01.lrit



IMG\_FD\_010\_IR105\_20211216\_014006\_01.lrit.jpg

- FD 세그먼트 10개의 이미지를 하나로 병합
- 각 이미지는 Indexed\_8 포맷으로 되어 있음  
(Indexed\_8: 한 화소(pixel)가 8bits)



여기까지 수행 시, FD 자료 수신 완료  
추가로 각 화소의 위경도, 밝기온도 계산 가능  
Header Type: #2 Image Navigation 참고  
Header Type: #3 Image Data Function 참고

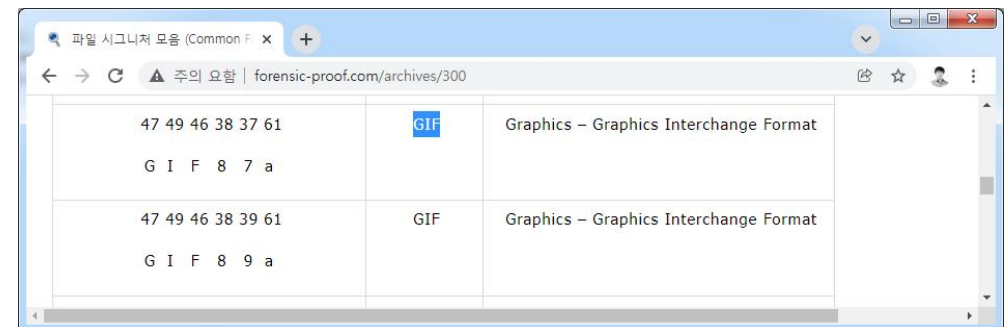
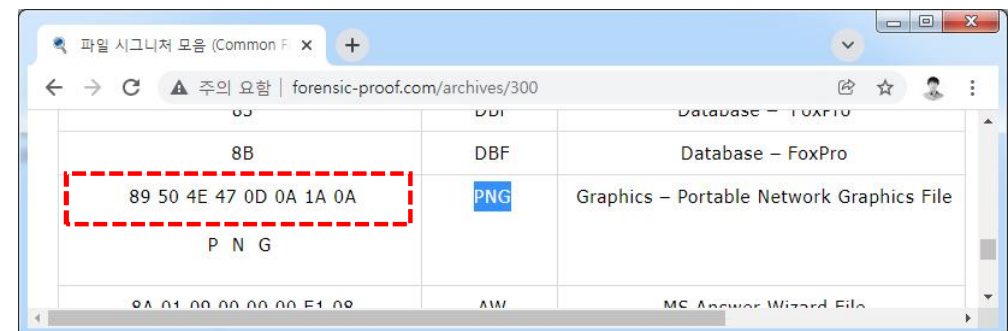
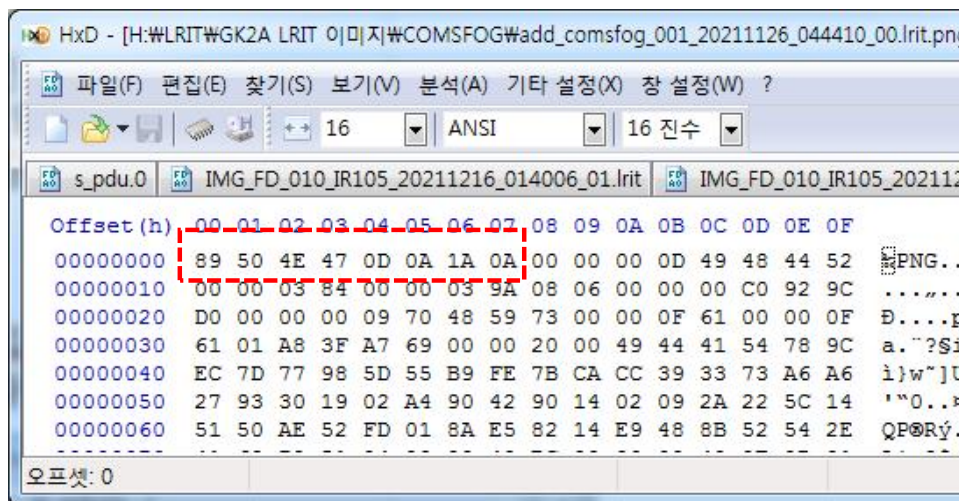
## Additional File Type의 LRIT 파일 처리

29 페이지

- Headers에서 파일명 확인
- Headers에서 암호화 여부 확인
- 이 자료는 일반적으로 암호화 되지 않음
- 일반적으로 단일 파일로 구성(segmentation X)
- 데이터 영역을 파일명으로 저장

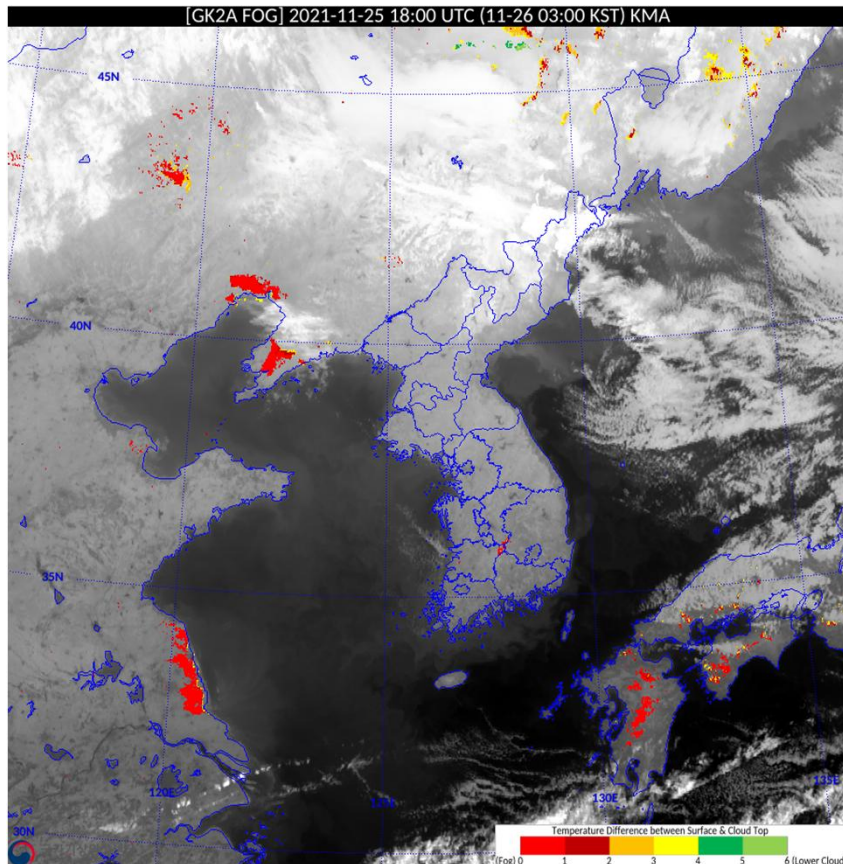
Headers	File Signature
...	데이터

예시) add\_comsfog\_001\_20211126\_044410\_00.lrit 파일을 수신했을 때, 확장자를 알기 위해서는 해당 파일의 시작점 조사 값이 89 50 4E로 시작한다면 add\_comsfog\_001\_20211126\_044410\_00.lrit.png로 저장





- 결과물 예시



여기까지 수행 시, ADD 자료 수신 완료  
ADD 자료는 별도의 추가 정보가 없음  
(단순 이미지, 텍스트 또는 음성 파일 전송)

add\_comsfog\_001\_20211126\_044410\_00.lrit.png

- ❖ 비트 연산 예시(VCDU에서 S/C ID 읽기)
- ❖ De-randomization Look-up 테이블 계산
- ❖ De-randomization 처리
- ❖ Reed-Solomon Decoding
- ❖ Source Packet의 CRC16체크
- ❖ 압축 해제

## 비트 연산 예시(VCDU에서 S/C ID 읽기)

- 읽고자 하는 값이 vcd�[0]~vcd�[1] 에 걸쳐 있는 상황
- vcd�[0]에서 6비트를 읽고, vcd�[1]에서 2비트를 읽어서 합쳐야 함.

Version #	VCDU-ID		VCDU Counter	Signaling Field	
	S/C ID	VCID		Replay Flag	Spare
2 bits	8 bits	6 bits	24 bits	1 bit	7 bits

vcd�[0]	vcd�[1]
0111 0000	1100 0000

vcd�[0] 에서 뒤의 6비트 읽기

1~6번 비트자리에 있어야 하므로

0011 0000    << 2  
1100 0000

vcd�[1] 에서 앞의 2비트 읽기

7, 8번 비트자리에 있어야 하므로

1100 0000    >> 6  
0000 0011

두 값 합치기

1100 0000  
OR 0000 0011  
-----  
1100 0011

```
unsigned char SCID = (vcd�[0] << 2) | (vcd�[1] >> 6);
```



- CCSDS TM Synchronization and channel coding 68p

## 10.4 SEQUENCE SPECIFICATION

**10.4.1** The pseudo-random sequence shall be generated using the following polynomial:

$$h(x) = x^8 + x^7 + x^5 + x^3 + 1$$

**10.4.2** This sequence shall begin at the first bit of the codeblock, codeword, or Transfer Frame and shall repeat after 255 bits, continuing repeatedly until the end of the codeblock,

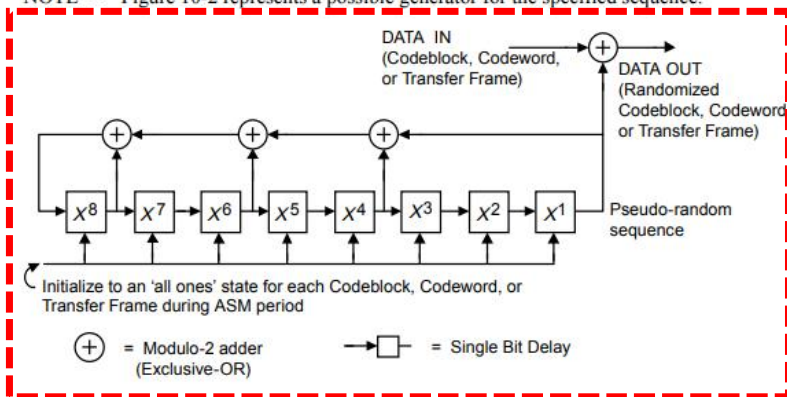
codeword, or Transfer Frame. The sequence generator shall be initialized to the all-ones state at the start of each codeblock, codeword, or Transfer Frame.

NOTE – The first 40 bits of the pseudo-random sequence from the generator are shown below. The leftmost bit is the first bit of the sequence to be exclusive-ORed with the first bit of the codeblock, codeword, or Transfer Frame; the second bit of the sequence is exclusive-ORed with the second bit of the codeblock, codeword, or Transfer Frame, and so on.

1111 1111 0100 1000 0000 1110 1100 0000 1001 1010 . . .

### 10.5 LOGIC DIAGRAM

NOTE – Figure 10-2 represents a possible generator for the specified sequence.



**Figure 10-2: Pseudo-Randomizer Logic Diagram**

XOR  
(배타적 논리합)

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

	XOR	XOR	XOR						
	8	7	6	5	4	3	2	1	
	1	1	1	1	1	1	1	1	
1	0	1	1	1	1	1	1	1	1
2	1	0	1	1	1	1	1	1	1
3	0	1	0	1	1	1	1	1	1
4	0	0	1	0	1	1	1	1	1
5	1	0	0	1	0	1	1	1	1
6	0	1	0	0	1	0	1	1	1
7	0	0	1	0	0	1	0	1	1
8	0	0	0	1	0	0	1	0	1
1	0	0	0	0	1	0	0	1	0
2	0	0	0	0	0	1	0	0	1
3	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	1	0
5	1	0	0	0	0	0	0	0	1
6	1	1	0	0	0	0	0	0	0
7	1	1	1	0	0	0	0	0	0
8	0	1	1	1	0	0	0	0	0
1	1	0	1	1	1	0	0	0	0
2	1	1	0	1	1	1	0	0	0
3	0	1	1	0	1	1	1	0	0
4	0	0	1	1	0	1	1	1	0
5	0	0	0	1	1	0	1	1	1
6	0	0	0	0	1	1	0	1	1
7	0	0	0	0	0	1	1	0	1
8	0	0	0	0	0	0	1	1	0

## De-randomization 처리

- 적용 방법

1. ASFM을 제외한 1020bytes에 대해
2. 1byte 단위로 XOR 수행

```
//De-Randomization
for (int i = 0; i < 1020; i++)
{
    cadu[i + 4] ^= pn[i % 255];
}
```

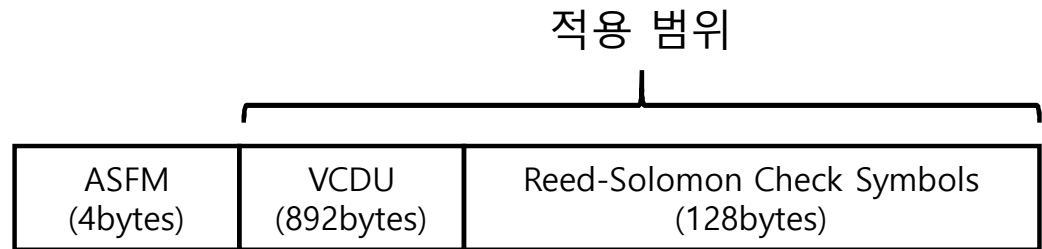
- 수행 결과 검증 방법

- VCDU 자료 헤더 출현여부 확인

```
unsigned char pn[255] = {
    0xff, 0x48, 0x0e, 0xc0, 0x9a, 0x0d, 0x70, 0xbc,
    0x8e, 0x2c, 0x93, 0xad, 0xa7, 0xb7, 0x46, 0xce,
    0x5a, 0x97, 0x7d, 0xcc, 0x32, 0xa2, 0xbf, 0x3e,
    0x0a, 0x10, 0xf1, 0x88, 0x94, 0xcd, 0xea, 0xb1,
    0xfe, 0x90, 0x1d, 0x81, 0x34, 0x1a, 0xe1, 0x79,
    0x1c, 0x59, 0x27, 0x5b, 0x4f, 0x6e, 0x8d, 0x9c,
    0xb5, 0x2e, 0xfb, 0x98, 0x65, 0x45, 0x7e, 0x7c,
    0x14, 0x21, 0xe3, 0x11, 0x29, 0x9b, 0xd5, 0x63,
    0xfd, 0x20, 0x3b, 0x02, 0x68, 0x35, 0xc2, 0xf2,
    0x38, 0xb2, 0x4e, 0xb6, 0x9e, 0xdd, 0x1b, 0x39,
    0x6a, 0x5d, 0xf7, 0x30, 0xca, 0x8a, 0xfc, 0xf8,
    0x28, 0x43, 0xc6, 0x22, 0x53, 0x37, 0xaa, 0xc7,
    0xfa, 0x40, 0x76, 0x04, 0xd0, 0x6b, 0x85, 0xe4,
    0x71, 0x64, 0x9d, 0x6d, 0x3d, 0xba, 0x36, 0x72,
    0xd4, 0xbb, 0xee, 0x61, 0x95, 0x15, 0xf9, 0xf0,
    0x50, 0x87, 0x8c, 0x44, 0xa6, 0x6f, 0x55, 0x8f,
    0xf4, 0x80, 0xec, 0x09, 0xa0, 0xd7, 0x0b, 0xc8,
    0xe2, 0xc9, 0x3a, 0xda, 0x7b, 0x74, 0x6c, 0xe5,
    0xa9, 0x77, 0xdc, 0xc3, 0x2a, 0x2b, 0xf3, 0xe0,
    0xa1, 0x0f, 0x18, 0x89, 0x4c, 0xde, 0xab, 0x1f,
    0xe9, 0x01, 0xd8, 0x13, 0x41, 0xae, 0x17, 0x91,
    0xc5, 0x92, 0x75, 0xb4, 0xf6, 0xe8, 0xd9, 0xcb,
    0x52, 0xef, 0xb9, 0x86, 0x54, 0x57, 0xe7, 0xc1,
    0x42, 0x1e, 0x31, 0x12, 0x99, 0xbd, 0x56, 0x3f,
    0xd2, 0x03, 0xb0, 0x26, 0x83, 0x5c, 0x2f, 0x23,
    0x8b, 0x24, 0xeb, 0x69, 0xed, 0xd1, 0xb3, 0x96,
    0xa5, 0xdf, 0x73, 0x0c, 0xa8, 0xaf, 0xcf, 0x82,
    0x84, 0x3c, 0x62, 0x25, 0x33, 0x7a, 0xac, 0x7f,
    0xa4, 0x07, 0x60, 0x4d, 0x06, 0xb8, 0x5e, 0x47,
    0x16, 0x49, 0xd6, 0xd3, 0xdb, 0xa3, 0x67, 0x2d,
    0x4b, 0xbe, 0xe6, 0x19, 0x51, 0x5f, 0x9f, 0x05,
    0x08, 0x78, 0xc4, 0x4a, 0x66, 0xf5, 0x58
};
```

- 전송 자료(892bytes)와 오류를 정정하기 위한 추가 128바이트 자료로 구성
- RS(255, 223, 4)로 코딩 시 한 VCDU는 최대 16\*4 바이트 개수의 에러까지 정정 가능
- 라이브러리 제공
- 적용 방법

```
CReedSolomon rs;  
rs.init(8, 16, 112, 11, 0, 4, 0, 1);  
rs.Decode(cadu + 4);  
  
rs.UncorrectableErrorsInFrame(); // 정정 불가  
rs.CorrectableErrorsInFrame(); // 정정된 오류 수
```



- 출력 데이터 검증
  - 오류가 없는 자료였다면 수행 결과 전후 자료에 변화가 없음
  - 수행 결과 값은 VCDU 형태가 되어야 함.

- 헤더를 제외한 데이터 영역의 오류 검사
- 오류 정정 불가, 오류 포함 여부만 확인 가능

```
bool SourcePacket::crc16()
{
    unsigned short myCRC;
    unsigned char* d = (unsigned char*)&myCRC;
    d[0] = dataField()[dataFieldSize()+1];
    d[1] = dataField()[dataFieldSize()+0];

    if (myCRC != CCITT_CRC16(dataField(), dataFieldSize()))
    {
        qWarning() << "CRC ERROR";
        return false;
    }

    return true;
}

unsigned short SourcePacket::CCITT_CRC16(const unsigned char *data_p, unsigned short len)
{
    unsigned int i;
    unsigned short vcrc;

    vcrc = 0xFFFF;

    for (i = 0; i < len; i++)
    {
        vcrc = (vcrc << 8) ^ CCITT_table[(vcrc >> 8) ^ data_p[i] & 0x000000FF];
    }

    return vcrc;
}
```

```
unsigned short CCITT_table[256] = {
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6,
    0x70E7, 0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD,
    0xE1CE, 0xF1EF, 0x1231, 0x0210, 0x3273, 0x2252, 0x52B5,
    0x4294, 0x72F7, 0x62D6, 0x9339, 0x8318, 0xB37B, 0xA35A,
    0xD3BD, 0xC39C, 0xF3FF, 0xE3DE, 0x2462, 0x3443, 0x0420,
    0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485, 0xA56A, 0xB54B,
    0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D, 0x3653,
    0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,
    0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D,
    0xC7BC, 0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861,
    0x2802, 0x3823, 0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948,
    0x9969, 0xA90A, 0xB92B, 0x5AF5, 0x4AD4, 0x7AB7, 0x6A96,
    0x1A71, 0x0A50, 0x3A33, 0x2A12, 0xDBFD, 0xCBDC, 0xFBBF,
    0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A, 0x6CA6, 0x7C87,
    0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41, 0xEDAE,
    0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,
    0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51,
    0x0E70, 0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A,
    0x9F59, 0x8F78, 0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C,
    0xC12D, 0xF14E, 0xE16F, 0x1080, 0x00A1, 0x30C2, 0x20E3,
    0x5004, 0x4025, 0x7046, 0x6067, 0x83B9, 0x9398, 0xA3FB,
    0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E, 0x02B1, 0x1290,
    0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256, 0xB5EA,
    0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,
    0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424,
    0x4405, 0xA7DB, 0xB7FA, 0x8799, 0x9778, 0xE75F, 0xF77E,
    0xC71D, 0xD73C, 0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657,
    0x7676, 0x4615, 0x5634, 0xD94C, 0xC96D, 0xF90E, 0xE92F,
    0x99C8, 0x89E9, 0xB98A, 0xA9AB, 0x5844, 0x4865, 0x7806,
    0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3, 0xCB7D, 0xDB5C,
    0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A, 0x4A75,
    0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,
    0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8,
    0x8DC9, 0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83,
    0x1CE0, 0x0CC1, 0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B,
    0xBFBA, 0x8FD9, 0x9FF8, 0x6E17, 0x7E36, 0x4E55, 0x5E74,
    0x2E93, 0x3EB2, 0x0ED1, 0x1EF0 };
```

- 라이브러리 제공
- 사용 예제

```
QFile file("D:/projects/test/LRITIngestor/Decrypted_lrit_data/IMG_FD_010_IR105_20211216_014006_05_dataArea.lrit");
file.open(QIODevice::ReadOnly);
QByteArray d = file.readAll();
file.close();

unsigned long long key = 0x3B077991????????; // 암호키 추후 제공
DES des;
des.setKey(key);
des.decrypt((unsigned char*)d.data(), d.size());

QFile file2(" D:/projects/test/LRITIngestor/Decrypted_lrit_data/IMG_FD_010_IR105_20211216_014006_05_dataArea.lrit.jpg");
file2.open(QIODevice::WriteOnly);
file2.write(d);
file2.close();
```

- 일반적인 jpg 파일이므로 대부분의 언어에서 jpg파일을 읽을 수 있음.
- Image Format은 Indexed\_8

```
QByteArray LRITFile::decompress()
{
    if (_imageStructure.Exists)
    {
        if (_imageStructure.Compression_Flag == 2)
        {
            QImage img;
            if (img.loadFromData(data(), dataLength()))
            {
                return QByteArray((char*)img.bits(), img.width() * img.height());
            }
        }
    }

    return QByteArray();
}
```

1. 개발 도구 및 언어(본인이 선택, C++ 권장 )
2. CADU 파일
3. 암호해제키
4. Derandomizer PN Table
5. Reed-Solomon Decoder 소스코드(\*.h, \*.cpp)
6. DES 소스코드(\*h, \*.cpp)

실시간 획득한 CADU 자료는 경우에 따라 에러가 포함 될 수 있음.  
에러가 포함된 자료로 처리 시, 예외 상황을 꼼꼼히 처리하지 않으면 프로그램 오류 발생  
RS decoding 결과에서 에러가 탐지되지 않은 자료로 실습할 것을 권함.

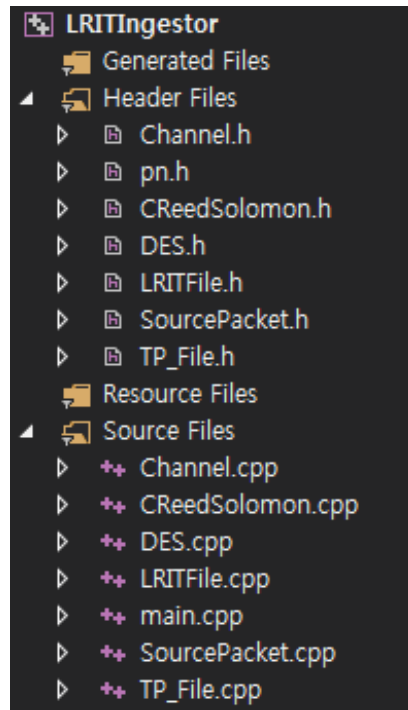
※ 단, Fill CADU 영역에서는 에러 발생해도 무관



- ❖ 라이브러리 추가
- ❖ CADU 읽기
- ❖ Channel Decoding
- ❖ VCDU를 각 채널로 출력
- ❖ 각 채널에서 VCDU를 소스패킷의 입력으로 출력
- ❖ 소스 패킷 헤더 읽기
- ❖ 소스 패킷을 TP\_파일로 조립
- ❖ TP\_File 헤더 읽기 및 검증
- ❖ LRIT 파일 헤더 읽기
- ❖ Big Endian to Little Endian 변환하여 읽기

여기에 소개된 샘플 소스 코드는 오류가 포함된 데이터를 입력할 경우 정상적으로 작동하지 않을 수 있음

## 1. 라이브러리 추가하기



## 2. CADU 읽기

```
int main(int argc, char *argv[])
{
    QFile file("d:/lrit_dump.cadu");
    if (file.open(QIODevice::ReadOnly) == false)
    {
        qDebug() << "No file found.";
        return -1;
    }

    char cadu[1024];
    while (file.atEnd() == false)
    {
        int readSize = file.read(cadu, 1024);

        if (readSize != 1024)
        {
            qDebug() << "End of file.";
            ::exit(0);
        }

        if ( false ) // Frame Sync 생략, check 0x1ACFFC1D
        {
            qDebug() << "Not found: ASFM";
            return -1;
        }

        ingest((unsigned char*)cadu);
    }

    file.close();
    qDebug() << "Processing completed.";

    return 0;
}
```

## 3. Channel Decoding

43 페이지

```
void derandomize(unsigned char* cadu)
{
    for (int i = 0; i < 1020; i++) {
        cadu[i + 4] ^= pn[i % 255];
    }
}

void rsdecode(unsigned char* cadu)
{
    CR ReedSolomon rsCodec;
    rsCodec.init(8, 16, 112, 11, 0, 4, 0, 1);
    rsCodec.Decode(cadu + 4);

    if (rsCodec.UncorrectableErrorsInFrame())
    {
        qDebug() << "RS ERROR FOUND";
    }
}

void ingest(unsigned char* cadu)
{
    derandomize(cadu);
    rsdecode(cadu);

    unsigned char* vcdu = cadu + 4;
    ingestVCDU(vcdu, 892);
}
```

## 4. VCDU를 각 채널로 출력

```
Channel channel63(63);
Channel channel0(0);
Channel channel4(4);
Channel channel5(5);

void ingestVCDU(unsigned char* vcd�, int size)
{
    unsigned char version = vcd�[0] >> 6;
    unsigned char SCID = (vcd�[0] << 2) | (vcd�[1] >> 6);
    unsigned char VCID = vcd�[1] & 0x3F;
    unsigned int vcd�Counter = (vcd�[2] << 16) + (vcd�[3] << 8) + vcd�[4];

    if (version == 1 && SCID == 0xC3)
    {
        if (VCID == 63) // fill cadu
        {
            return;
        }
        else if (VCID == 0) // Image File
        {
            channel0.appendToSourcePacket(vcd�Counter, vcd� + 6, 886);
            return;
        }
        else if (VCID == 4) //AlphaNumeric Text
        {
            channel4.appendToSourcePacket(vcd�Counter, vcd� + 6, 886);
            return;
        }
        else if (VCID == 5) // Additional Data
        {
            channel5.appendToSourcePacket(vcd�Counter, vcd� + 6, 886);
            return;
        }
    }
    qDebug() << "Invalid packet(version scid vcid)" << version << SCID << VCID;
}
```

## 5. 각 채널에서 VCDU를 소스패킷의 입력으로 출력

```
void Channel::appendToSourcePacket(unsigned int voidCounter, unsigned char* data, int datasize)
{
    int spare = data[0] >> 3;
    int firstHeaderPointer = ((data[0] & 0x1F) << 8) | data[1];
    unsigned char* _pPacketZone = data + 2;

    if (firstHeaderPointer == 0x07FF || voidCounter == 0 )
    {
        _sourcePacket->append(_pPacketZone, 884);
    }
    else
    {
        _sourcePacket->append(_pPacketZone, firstHeaderPointer);
        _sourcePacket->onCompleted();

        appendToTPFile(_sourcePacket);

        _sourcePacket = new SourcePacket();
        _sourcePacket->append(_pPacketZone + firstHeaderPointer, 884 - firstHeaderPointer);
    }
}

void Channel::appendToTPFile(SourcePacket* pSourcePacket)
{
    if (_sourcePacket->isValid())
    {
        _tpFile.append(pSourcePacket);
    }
    else
    {
        delete pSourcePacket;
        pSourcePacket = NULL;
    }
}
```

## 6. 소스 패킷 헤더 읽기

```
void SourcePacket::append(unsigned char* data, int size)
{
    if (_lastPos + size <= 8198)
    {
        memcpy(_data + _lastPos, data, size);
    }
    else
    {
        // error
    }
    _lastPos += size;
}

void SourcePacket::onCompleted()
{
    readHeader();
}
```

```
bool SourcePacket::readHeader()
{
    if (_lastPos > 8) // Header 6 + CRC 2 + Data 1 at least
    {
        _version = (_data[0] & 0xE0) >> 5;
        _type = (_data[0] & 0x10) >> 4;
        _secondHeaderFlag = (_data[0] & 0x08) >> 3;
        _apid = ((_data[0] & 0x03) << 8) + _data[1];
        _sequenceFlag = (_data[2] & 0xC0) >> 6;
        _packetSequenceCount = ((_data[2] & 0x3F) << 8) + _data[3];
        _packetLength = (_data[4] << 8) + _data[5];

        if (_packetLength)
        {
            if (crc16() == false)
            {
                return false;
            }

            if (_packetLength + 7 == _lastPos)
            {
                _validated = true;
                return true;
            }
        }
        return false;
    }
}

const unsigned char* SourcePacket::dataField()
{
    return (_data + 6);
}

int SourcePacket::dataFieldSize()
{
    return _packetLength - 1;
}
```

## 7. 소스 패킷을 TP\_파일로 조립

```
void TP_File::append(SourcePacket* pSourcePacket)
{
    if (pSourcePacket->_sequenceFlag == 1) // (first segment)
    {
        initBuffer();
        _sourcePackets += pSourcePacket;
    }

    else if (pSourcePacket->_sequenceFlag == 0) // (continued segment)
    {
        _sourcePackets += pSourcePacket;
    }

    else if (pSourcePacket->_sequenceFlag == 2) // (last segment)
    {
        _sourcePackets += pSourcePacket;
        makeTPFile();
        initBuffer();
    }
    else if (pSourcePacket->_sequenceFlag == 3) // (single data)
    {
        initBuffer();
        makeTPFile();
        initBuffer();
    }
    else
    {
        // Not defined _sequenceFlag
    }
}

void TP_File::initBuffer()
{
    for (int i = 0; i < _sourcePackets.count(); i++)
    {
        delete _sourcePackets[i];
        _sourcePackets[i] = NULL;
    }
}
```

## 8. TP\_File 헤더 읽기 및 검증

47 페이지

```
void TP_File::makeTPFile()
{
    unsigned int tp_file_size = getSegmentFileSize();
    unsigned char* tp_file = new unsigned char[tp_file_size];

    int lastPos = 0;
    for (int i = 0; i < _sourcePackets.count(); i++)
    {
        memcpy(tp_file + lastPos, _sourcePackets[i]->dataField(), _sourcePackets[i]->dataFieldSize());
        lastPos += _sourcePackets[i]->dataFieldSize();
    }

    int fileCounter = (tp_file[0] << 8) + tp_file[1];
    unsigned long long fileLengthInBits = 0;
    fileLengthInBits += (unsigned long long) tp_file[2] << 56;
    fileLengthInBits += (unsigned long long) tp_file[3] << 48;
    fileLengthInBits += (unsigned long long) tp_file[4] << 40;
    fileLengthInBits += (unsigned long long) tp_file[5] << 32;
    fileLengthInBits += tp_file[6] << 24;
    fileLengthInBits += tp_file[7] << 16;
    fileLengthInBits += tp_file[8] << 8;
    fileLengthInBits += tp_file[9];

    unsigned int s_pdu_size = fileLengthInBits / 8;
    if ((s_pdu_size + 10) == tp_file_size)
    {
        onS_PDU_Created(tp_file + 10, s_pdu_size); // LRIT File
    }

    delete[] tp_file;
    tp_file = NULL;
}
```

## 9. LRIT 파일 헤더 읽기

```
bool LRITFile::readHeader()
{
    if (_fileSize < 16) // 헤더 길이는 최소한 16바이트 이상이어야 함
        return false;

    do
    {
        unsigned HeaderType = readUChar();
        unsigned HeaderRecordLength = readUShort();

        if (HeaderType == 0) // Header Type#0 - Primary Header
        {
            _primaryHeader.Exists = true;
            _primaryHeader.HeaderType = HeaderType;
            _primaryHeader.HeaderRecordLength = HeaderRecordLength;
            _primaryHeader.FileTypeCode = readUChar();
            _primaryHeader.TotalHeaderLength = readUInt();
            _primaryHeader.DataFieldLength = readULongLong();
        }
        else if (HeaderType == 1) // Header Type#1 - Image Structure
        {
            _imageStructure.Exists = true;
            _imageStructure.HeaderType = HeaderType;
            _imageStructure.HeaderRecordLength = HeaderRecordLength;
            _imageStructure.NumberOfBitPerPixel = readUChar();
        }
        else if (HeaderType == 7) // Header Type#7 - Key header
        {
            _keyHeader.Exists = true;
            _keyHeader.HeaderType = HeaderType;
            _keyHeader.HeaderRecordLength = HeaderRecordLength;
            _keyHeader.Key_Number = readUInt();
        }
        else
        {
            qDebug() << "Need to implement Header Type" << HeaderType;
            break;
        }
    } while (_pHeaderPos < (_primaryHeader.TotalHeaderLength + _pFileStartPos));

    return _fileSize == (_primaryHeader.TotalHeaderLength + _primaryHeader.DataFieldLength / 8);
}
```

중략

## 10. Big Endian to Little Endian 변환하여 읽기 48 페이지

```
unsigned char LRITFile::readUChar()
{
    unsigned char v;
    v = _pHeaderPos[0];
    _pHeaderPos += 1;
    return v;
}

unsigned short LRITFile::readUShort()
{
    unsigned short v;
    char* p = (char*)&v;
    p[0] = _pHeaderPos[1];
    p[1] = _pHeaderPos[0];

    _pHeaderPos += 2;
    return v;
}

unsigned int LRITFile::readUInt()
{
    unsigned int v;
    char* p = (char*)&v;
    p[0] = _pHeaderPos[3];
    p[1] = _pHeaderPos[2];
    p[2] = _pHeaderPos[1];
    p[3] = _pHeaderPos[0];

    _pHeaderPos += 4;
    return v;
}

int LRITFile::readInt()
{
    int v;
    char* p = (char*)&v;
    p[0] = _pHeaderPos[3];
    p[1] = _pHeaderPos[2];
    p[2] = _pHeaderPos[1];
    p[3] = _pHeaderPos[0];
}
```



## 11. LRIT 파일 저장

49 페이지

```
void LRITFile::save(QString filePath)
{
    QFile file(filePath);
    file.open(QIODevice::WriteOnly);
    file.write((char*)_pFileStartPos, _fileSize);
    file.close();
}



































void LRITFile::saveContentFile(QString filePath)
{
    decrypt();

    QByteArray suffix = getSuffix();
    QFile file(filePath + "." + suffix);
    file.open(QIODevice::WriteOnly);
    file.write((char*)data(), dataLength());
    file.close();
}

QByteArray LRITFile::getSuffix()
{
    unsigned char* p = data();

    if (p[0] == 0xFF && p[1] == 0x08)
    {
        return QByteArray("jpg");
    }
    else if (p[0] == 0x89 && p[1] == 0x50 && p[2] == 0x4E)
    {
        return QByteArray("png");
    }
    else if (p[0] == 0x47 && p[1] == 0x49 && p[2] == 0x46)
    {
        return QByteArray("gif");
    }

    return QByteArray("txt");
}
```

 ADD_RWW3A_001_20211216_014300_00.lrit	2022-01-12 오후 4:39	LRIT 파일	91KB
 ADD_RWW3A_001_20211216_014300_00.lrit.gif	2022-01-12 오후 4:39	알씨 GIF 파일	91KB
 IMG_FD_010_JR105_20211216_014006_02.lrit	2022-01-12 오후 4:39	LRIT 파일	67KB
 IMG_FD_010_JR105_20211216_014006_02.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	67KB
 IMG_FD_010_JR105_20211216_014006_03.lrit	2022-01-12 오후 4:39	LRIT 파일	72KB
 IMG_FD_010_JR105_20211216_014006_03.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	72KB
 IMG_FD_010_JR105_20211216_014006_04.lrit	2022-01-12 오후 4:39	LRIT 파일	91KB
 IMG_FD_010_JR105_20211216_014006_04.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	91KB
 IMG_FD_010_JR105_20211216_014006_05.lrit	2022-01-12 오후 4:39	LRIT 파일	122KB
 IMG_FD_010_JR105_20211216_014006_05.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	122KB
 IMG_FD_010_JR105_20211216_014006_06.lrit	2022-01-12 오후 4:39	LRIT 파일	132KB
 IMG_FD_010_JR105_20211216_014006_06.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	132KB
 IMG_FD_010_JR105_20211216_014006_07.lrit	2022-01-12 오후 4:39	LRIT 파일	95KB
 IMG_FD_010_JR105_20211216_014006_07.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	95KB
 IMG_FD_010_JR105_20211216_014006_08.lrit	2022-01-12 오후 4:39	LRIT 파일	94KB
 IMG_FD_010_JR105_20211216_014006_08.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	94KB
 IMG_FD_010_JR105_20211216_014006_09.lrit	2022-01-12 오후 4:39	LRIT 파일	80KB
 IMG_FD_010_JR105_20211216_014006_09.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	80KB
 IMG_FD_010_JR105_20211216_014006_10.lrit	2022-01-12 오후 4:39	LRIT 파일	42KB
 IMG_FD_010_JR105_20211216_014006_10.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	42KB
 IMG_FD_011_JR105_20211216_015006_02.lrit	2022-01-12 오후 4:39	LRIT 파일	67KB
 IMG_FD_011_JR105_20211216_015006_02.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	67KB
 IMG_FD_011_JR105_20211216_015006_03.lrit	2022-01-12 오후 4:39	LRIT 파일	73KB
 IMG_FD_011_JR105_20211216_015006_03.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	73KB
 IMG_FD_011_JR105_20211216_015006_04.lrit	2022-01-12 오후 4:39	LRIT 파일	91KB
 IMG_FD_011_JR105_20211216_015006_04.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	91KB
 IMG_FD_011_JR105_20211216_015006_05.lrit	2022-01-12 오후 4:39	LRIT 파일	122KB
 IMG_FD_011_JR105_20211216_015006_05.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	122KB
 IMG_FD_011_JR105_20211216_015006_06.lrit	2022-01-12 오후 4:39	LRIT 파일	132KB
 IMG_FD_011_JR105_20211216_015006_06.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	132KB
 IMG_FD_011_JR105_20211216_015006_07.lrit	2022-01-12 오후 4:39	LRIT 파일	95KB
 IMG_FD_011_JR105_20211216_015006_07.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	95KB
 IMG_FD_011_JR105_20211216_015006_08.lrit	2022-01-12 오후 4:39	LRIT 파일	94KB
 IMG_FD_010_JR105_20211216_014006_01.lrit.jpg	2022-01-12 오후 4:39	알씨 JPG 파일	38KB

FD자료의 경우 병합 필요

수고하셨습니다!